Advances in Artificial General Intelligence: Concepts, Architectures and Algorithms

Proceedings of the AGI Workshop 2006

Edited by

Ben Goertzel

Novamente LLC

and

Pei Wang

Temple University

IOS Press

Amsterdam • Berlin • Oxford • Tokyo • Washington, DC

VISIT…

LANZAROTE Caliente.COM

Frontiers in Artificial Intelligence and Applications

FAIA covers all aspects of theoretical and applied artificial intelligence research in the form of monographs, doctoral dissertations, textbooks, handbooks and proceedings volumes. The FAIA series contains several sub-series, including “Information Modelling and Knowledge Bases” and “Knowledge-Based Intelligent Engineering Systems”. It also includes the biennial ECAI, the European Conference on Artificial Intelligence, proceedings volumes, and other ECCAI – the European Coordinating Committee on Artificial Intelligence – sponsored publications. An editorial panel of internationally well-known scholars is appointed to provide a high quality selection.

Series Editors:
J. Breuker, R. Dieng-Kuntz, N. Guarino, J.N. Kok, J. Liu, R. López de Mántaras,
R. Mizoguchi, M. Musen and N. Zhong

Volume 157

Recently published in this series

Vol. 156. R.M. Colomb, Ontology and the Semantic Web
Vol. 155. O. Vasilecas et al. (Eds.), Databases and Information Systems IV – Selected Papers from the Seventh International Baltic Conference DB&IS’2006
Vol. 154. M. Duží et al. (Eds.), Information Modelling and Knowledge Bases XVIII
Vol. 153. Y. Vogiazou, Design for Emergence – Collaborative Social Play with Online and Location-Based Media
Vol. 152. T.M. van Engers (Ed.), Legal Knowledge and Information Systems – JURIX 2006: The Nineteenth Annual Conference
Vol. 151. R. Mizoguchi et al. (Eds.), Learning by Effective Utilization of Technologies: Facilitating Intercultural Understanding
Vol. 150. B. Bennett and C. Fellbaum (Eds.), Formal Ontology in Information Systems – Proceedings of the Fourth International Conference (FOIS 2006)
Vol. 149. X.F. Zha and R.J. Howlett (Eds.), Integrated Intelligent Systems for Engineering Design
Vol. 148. K. Kersting, An Inductive Logic Programming Approach to Statistical Relational Learning
Vol. 147. H. Fujita and M. Mejri (Eds.), New Trends in Software Methodologies, Tools and Techniques – Proceedings of the fifth SoMeT_06
Vol. 146. M. Polit et al. (Eds.), Artificial Intelligence Research and Development
Vol. 145. A.J. Knobbe, Multi-Relational Data Mining
Vol. 144. P.E. Dunne and T.J.M. Bench-Capon (Eds.), Computational Models of Argument – Proceedings of COMMA 2006

ISSN 0922-6389

© 2007 The authors and IOS Press.

All rights reserved. No part of this book may be reproduced, stored in a retrieval system,
or transmitted, in any form or by any means, without prior written permission from the publisher.

ISBN 978-1-58603-758-1
Library of Congress Control Number: 2007927655

Publisher
IOS Press
Nieuwe Hemweg 6B
1013 BG Amsterdam
Netherlands
fax: +31 20 687 0019
e-mail: order@iospress.nl

Distributor in the UK and Ireland
Gazelle Books Services Ltd.
White Cross Mills
Hightown
Lancaster LA1 4XS
United Kingdom
fax: +44 1524 63232
e-mail: sales@gazellebooks.co.uk

Distributor in the USA and Canada
IOS Press, Inc.
4502 Rachael Manor Drive
Fairfax, VA 22032
USA
fax: +1 703 323 3668
e-mail: iosbooks@iospress.com

LEGAL NOTICE
The publisher is not responsible for the use which might be made of the following information.

PRINTED IN THE NETHERLANDS

ADVANCES IN ARTIFICIAL GENERAL INTELLIGENCE: CONCEPTS, ARCHITECTURES AND ALGORITHMS

Advances in Artificial General Intelligence: Concepts, Architectures and Algorithms
B. Goertzel and P. Wang (Eds.)
IOS Press, 2007
© 2007 The authors and IOS Press. All rights reserved.

Preface

Ben GOERTZEL

The topic of this book – the creation of software programs displaying broad, deep, human-style general intelligence – is a grand and ambitious one. And yet it is far from a frivolous one: what the papers here illustrate is that it is a fit and proper subject for serious science and engineering exploration. No one has yet created a software program with human-style or (even roughly) human-level general intelligence – but we now have a sufficiently rich intellectual toolkit that it is possible to think about such a possibility in detail, and make serious attempts at design, analysis and engineering. This is the situation that led to the organization of the 2006 AGIRI (Artificial General Intelligence Research Institute) workshop; and to the decision to pull together a book from contributions by the speakers at the conference.

The themes of the book and the contents of the chapters are discussed in the Introduction by myself and Pei Wang; so in this Preface I will restrict myself to a few brief and general comments.

As it happens, this is the second edited volume concerned with Artificial General Intelligence (AGI) that I have co-edited. The first was entitled simply *Artificial General Intelligence*; it appeared in 2006 under the Springer imprimatur, but in fact most of the material in it was written in 2002 and 2003. It is interesting to compare the material contained in the present volume, which was written in 2006, with the material from the previous volume. What is striking in performing this comparison is the significant movement toward practical realization that has occurred in the intervening few years.

The previous volume contained some very nice mathematical theory (e.g. by Marcus Hutter and Juergen Schmidhuber) pertaining to AGI under assumptions of near-infinite computational resources, some theory about the nature of intelligence as pertaining to AGI, and some descriptions of practical AGI projects at fairly early stages of development (including the NARS and Novamente systems developed by Pei Wang and myself respectively). The current volume, on the other hand, seems to represent significant progress. To take just a few examples: In the current volume, there is theoretical work (Eric Baum's and Moshe Looks' papers) that takes up Hutter's and Schmidhuber's emphasis on algorithmic information, and ties it in with practical suggestions regarding near-term AI design. My own Novamente system, which was described in fairly abstract terms in the earlier volume, is here represented by several papers by various authors reporting specific mathematical and experimental results, concretizing some (though by no means all, yet!) of the speculations made in the paper on Novamente in the previous volume. And, here we have a sufficient number of AGI design proposals, depicted in sufficient detail, that we have considered it worthwhile to include a chapter specifically comparing and contrasting four of the designs presented herein (Novamente, NARS, and the designs proposed by Stan Franklin and Alexei Samsonovich in their chapters).

In sum, what seems evident in comparing the prior volume with this one is that, while the end goal of the AGI research programme has not yet been achieved (and the proximity of achievement remains difficult to objectively predict), the field is gradually broadening its scope beyond mathematical and conceptual ideas, and becoming more of a practical pursuit.

And I am confident that if there is another edited volume in another 2 or 3 years time, the field will appear yet further dramatically advanced. The “AGI Winter” is thawing, and the AI field is now finally making sensible progress toward its original goal of creating truly thinking machines. The material presented here only scratches the surface of the AGI-related R&D work that is occurring around the world at this moment. But I am pleased to have had the chance to be involved in organizing and presenting at least a small percentage of the contemporary progress.

Finally, thanks must be extended to those who helped this volume, and the workshop that inspired it, to come into being. Bruce Klein deserves the lion’s share of thanks, as the 2006 AGIRI Workshop would not have come into being without his extraordinary vision and dedication. Everyone who attended the workshop also deserves a piece of gratitude, and especially those who spoke or participated in panel discussions. Anya Kolonin did a fine job of reformatting the manuscript for publication. And finally I must extend heartfelt thanks to my co-editor Pei Wang for his part in helping to pull together this book, and the scientific program of the workshop that inspired it. In my work with him over the years I have found Pei to display a combination of good sense, insight and reliability that is distressingly rare in this world (populated as it is by mere humans ... for now...).

Contents

Advances in Artificial General Intelligence: Concepts, Architectures and Algorithms
B. Goertzel and P. Wang (Eds.)
IOS Press, 2007

© 2007 The authors and IOS Press. All rights reserved.

Introduction: Aspects of Artificial General Intelligence

Pei WANG and Ben GOERTZEL

Introduction

This book contains materials that come out of the Artificial General Intelligence Research Institute (AGIRI) Workshop, held in May 20-21, 2006 at Washington DC. The theme of the workshop is "Transitioning from Narrow AI to Artificial General Intelligence."

In this introductory chapter, we will clarify the notion of "Artificial General Intelligence", briefly survey the past and present situation of the field, analyze and refute some common objections and doubts regarding this area of research, and discuss what we believe needs to be addressed by the field as a whole in the near future. Finally, we will briefly summarize the contents of the other chapters in this collection.

1. What is meant by "AGI"

"Artificial General Intelligence", AGI for short, is a term adopted by some researchers to refer to their research field. Though not a precisely defined technical term, the term is used to stress the "general" nature of the desired capabilities of the systems being researched -- as compared to the bulk of mainstream Artificial Intelligence (AI) work, which focuses on systems with very specialized "intelligent" capabilities. While most existing AI projects aim at a certain aspect or application of intelligence, an AGI project aims at "intelligence" as a whole, which has many aspects, and can be used in various situations. There is a loose relationship between "general intelligence" as meant in the term AGI and the notion of "g-factor" in psychology [1]: the g-factor is an attempt to measure general intelligence, intelligence across various domains, in humans.

The notion of "intelligence" itself has no universally accepted definition, and the chapter following this one surveys a variety of definitions found in various parts of the research literature. So, "general intelligence" as we use it here is an imprecise variation on an imprecise concept. However, such imprecise concepts are what guide the direction of research, including research into the precise formulation of concepts. We believe that "general intelligence" and AGI are important concepts to pursue, in terms of both theory and software implementation.

Modern learning theory has made clear that the only way to achieve *maximally* general problem-solving ability is to utilize infinite computing power. Intelligence given limited computational resources is always going to have limits to its generality. The human mind/brain, while possessing extremely general capability, is best at solving the types of problems which it has specialized circuitry to handle (e.g. face recognition, social learning, language learning; see [2] for a summary of arguments in this regard). However, even though no real intelligence can display total generality, it still makes sense to distinguish systems with general scope from highly specialized systems like chess-playing programs and automobile navigation systems and medical diagnosis systems. It is possible to quantify this distinction in various ways (see [3] and [4]; [5]; [6], for example), and this sort of quantification is an active area of research in itself, but for our present purposes drawing the qualitative distinction will suffice.

An AGI system, when fully implemented, will very likely be similar to the human brain/mind in various senses. However, we do not impose this as part of the definition of AGI. Nor do we restrict the techniques to be used to realize general intelligence, which means that an AGI project can follow a symbolic, connectionist, evolutionary, robotic, mathematical, or integrative approach. Indeed, the works discussed in the following chapters are based on very different theoretical and technical foundations, while all being AGI, as far as the goal and scope of the research is concerned. We believe that at the current stage, it is too early to conclude with any scientific definiteness which conception of "intelligence" is the "correct" one, or which technical approach is most efficient for achieving such a goal. It will be necessary for the field to encourage different approaches, and leave their comparison and selection to individual researchers.

However, this inclusiveness regarding methodology does not mean that all the current AI research projects can be labeled as "AGI". Actually, we believe that most of them cannot, and that is why we favor the use of a new term to distinguish the research we are interested in from what is usually called "AI". Again, the key difference is the goal and scope of the research. For example, nobody has challenged the belief that "learning" plays an important role in intelligence, and therefore it is an issue that almost all AGI projects address. However, most of the existing "machine learning" works do not belong to AGI, as defined above, because they define the learning problem in isolation, without treating it as part of a larger picture. They are not concerned with creating a system possessing broad-scope intelligence with generality at least roughly equal to that of the human mind/brain; they are concerned with learning in much narrower contexts. Machine learning algorithms may be applied quite broadly in a variety of contexts, but the breadth and generality in this case is supplied largely by the human user of the algorithm; any particular machine learning program, considered as a holistic system taking in inputs and producing outputs without detailed human intervention, can solve only problems of a very specialized sort.

Specified in this way, what we call AGI is similar to some other terms that have been used by other authors, such as "strong AI" [7], "human-level AI" [8], "true synthetic intelligence" [9], "general intelligent system" [10], and even "thinking machine" [11]. Though no term is perfect, we chose to use "AGI" because it correctly stresses the *general* nature of the research goal and scope, without committing too much to any theory or technique.

We will also refer in this chapter to "AGI projects." We use this term to refer to an AI research project that satisfies all the following criteria:

1. The project is based on a theory about "intelligence" as a whole (which may encompass intelligence as displayed by the human brain/mind, or may specifically refer to a class of non-human-like systems intended to display intelligence with a generality of scope at least roughly equalling that of the human brain/mind).

2. There is an engineering plan to implement the above conception of intelligence in a computer system.
3. The project has already produced some concrete results, as publications or prototypes, which can be evaluated by the research community.

The chapters in this book describe a number of current AGI research projects, thus defined, and also present some AGI research ideas not tied to any particular project.

2. The past and present of AGI

A comprehensive overview of historical and contemporary AGI projects is given in the introductory chapter of a prior edited volume focused on Artificial General Intelligence [5]. So, we will not repeat that material here. Instead, we will restrict ourselves to discussing a few recent developments in the field, and making some related general observations.

What has been defined above as “AGI” is very similar to the original concept of “AI”. When the first-generation AI researchers started their exploration, they dreamed to eventually build computer systems with capabilities comparable to those of the human mind in a wide range of domains. In many cases, such a dream remained in their minds throughout their whole career, as evidenced for instance by the opinions of Newell and Simon [12]; [13], Minsky [14], and McCarthy [8]. And, at various points in the history of the field, large amounts of resources were invested into projects aimed at AGI, as exemplified by the Fifth Generation Computer Systems project.

However, in spite of some initial successes and the high expectations they triggered, the attempts of the first wave of AI researchers did not result in functional AGI systems. As a consequence, the AI community to a large extent has abandoned its original dream, and turned to more “practical” and “manageable” problems. After half a century, “AI has evolved to being a label on a family of relatively disconnected efforts” [9]. Though many domain-specific problems have been solved, and many special-purpose tools have been built, not many researchers feel that these achievements have brought us significantly closer to the goal of AGI. What makes the situation worse is the fact that AGI research is not even encouraged. For a long time, the “AI dream” was rarely mentioned within the AI community, and whoever pursued it was effectively committing career suicide, since few people took such attempts seriously.

In recent years, several forces seem to be turning this unfortunate trend around.

First, the year of 2006 is the fiftieth anniversary of the AI discipline, and 2005 the twenty-fifth anniversary of AAAI. Many people have taken this opportunity to reassess the field, surveying its past, present, and future. Among all the voices, a recurring one calls for the field to return to its original goal [8]; [9]; [15].

Second, some long-term AGI projects have survived and made progress. For example, Cyc recently released its open source version; Soar has been adding functionality into the system, and extending its application domain. Though each of these techniques has its limitations, they nevertheless show that AGI research can be fruitful.

Finally, after decades of work at the margin or outside of the AI community, a new generation of AGI projects has matured to the extent of producing publications and preliminary results. More or less coincidentally, several books have appeared within the last few years, presenting several AGI projects, with theoretical and technical designs with various levels of detail [2]; [16]; [3]; [17]; [5]; [6]. Though each of these projects points to a quite different direction for AGI, they do introduce new ideas into the field, and show that the concrete, near-term possibilities of AGI are far from being exhaustively explored.

These factors have contributed to the recent resurgence of interest in AGI research. Only in the year of 2006, there have been several AGI-related gatherings in various conferences:

- Integrated Intelligent Capabilities (AAAI Special Track)
- A Roadmap to Human-Level Intelligence (IEEE WCCI Panel Session)
- Building and Evaluating Models of Human-Level Intelligence (CogSci Symposium)
- The AGIRI workshop, of which this book is a post-proceedings volume

Considering the fact that there were hardly any AGI-related meetings at all before 2004, the above list is quite remarkable.

3. Objections to AGI

Though the overall atmosphere is becoming more AGI-friendly, the researchers in this field remain a very small minority in the AI community. This situation is partially caused by various misunderstandings about AGI. As Turing did in [11], in the following paragraphs we will analyze and reject some common objections or doubts about AGI research.

3.1. "AGI is impossible"

Since the very beginning of AI research, there have been claims regarding the impossibility of truly intelligent computer systems. The best known arguments include those from Lucas [18], Dreyfus [19], and Penrose [20]. Since there is already a huge literature on these arguments [21], we will not repeat them here, but will simply remark that, so far, none of these arguments has convinced a majority of scientists with relevant expertise. Therefore, AGI remains possible, at least in theory.

3.2. "There is no such a thing as general intelligence"

There has been a lasting debate in the psychological research of human intelligence on whether there is a "general intelligence factor" ("g factor") that can be used to explain the difference in intellectual capabilities among individual human beings. Even though there is evidence supporting the existence of such a factor, as noted above there is also evidence suggesting that human intelligence is domain dependent, so is not that "general" at all.

In AI, many people have argued against ideas like the "General Problem Solver" [12], by stating that intelligent problem solving heavily depends on domain-specific knowledge. Guided by this kind of belief, various types of expert systems have been developed, whose power mostly come from domain knowledge, without which the system has little capability.

The above opinions do not rule out the possibility of AGI, for several reasons.

When we say a computer system is “general purpose”, we do not require a single factor in the system to be responsible for all of its cross-domain intelligence. It is possible for the system to be an integration of several techniques, so as to be general-purpose without a single g-factor.

Also, AGI does not exclude individual difference. It is possible to implement multiple copies of the same AGI design, with different parameters and innate capabilities, and the resulting systems grew into different “experts”, such as one with better mathematical capability, with another is better in verbal communication. Even in this case, the design is still “general” in the sense that it allows all these potentials. Just as the human brain/mind has a significant level of generality to its intelligence, even though some humans are better at mathematics and some are better at basketball.

Similarly, a general design does not conflict with the usage of domain-specific knowledge in problem solving. Even when an AGI system depends on domain-specific knowledge to solve domain-specific problems, its overall knowledge-management and learning mechanism may still remain general. The key point is that a general intelligence must be able to master a variety of domains, and learn to master new domains that it never confronted before. It does not need to have equal capability in all domains – humans will never be as intuitively expert at quantum physics as we are at the physics of projectiles in Earth’s atmosphere, for example. Our innate, domain-specific knowledge gives us a boost regarding the latter; but, our generality of intelligence allows us to handle the former as well, albeit slowly and awkwardly and with the frequent need of tools like pencils, paper, calculators and computers.

In the current context, when we say that the human mind or an AGI system is “general purpose”, we do not mean that it can solve all kinds of problems in all kinds of domains, but that it has the *potential* to solve any problem in any domain, given proper experience. Non-AGI systems lack such a potential. Even though Deep Blue plays excellent chess, it cannot do much other than that, no matter how it is trained.

3.3. “General-purpose systems are not as good as special-purpose ones”

Compared to the previous one, a weaker objection to AGI is to insist that even though general-purpose systems can be built, they will not work as well as special-purpose systems, in terms of performance, efficiency, etc.

We actually agree with this judgment to a certain degree, though we do not take it as a valid argument against the need to develop AGI.

For any given problem, a solution especially developed for it almost always works better than a general solution that covers multiple types of problem. However, we are not promoting AGI as a technique that will replace all existing domain-specific AI techniques. Instead, AGI is needed in situations where ready-made solutions are not available, due to the dynamic nature of the environment or the insufficiency of knowledge about the problem. In these situations, what we expect from an AGI system are not optimal solutions (which cannot be guaranteed), but flexibility, creatively, and robustness, which are directly related to the generality of the design.

In this sense, AGI is not proposed as a competing tool to any AI tool developed before, by providing better results, but as a tool that can be used when no other tool can, because the problem is unknown in advance.

3.4. "AGI is already included in the current AI"

We guess many AI researchers may be sympathetic to our goal, but doubt the need to introduce a new subfield into the already fragmented AI community. If what we call "AGI" is nothing but the initial and ultimate goal of AI, why bother to draw an unnecessary distinction?

We do this mostly for practical reasons, rather than theoretical reasons. Even though "AGI" is indeed closely related to the *original* meaning of "AI", so that it is in a sense a new word for an old concept, it is still very different from the *current* meaning of "AI", as the term is used in conferences and publications. As mentioned previously, our observation is that the mainstream AI community has been moving away from the original goal for decades, and we do not expect the situation to change completely very soon.

We do not buy the argument that "Since X plays an important role in intelligence, studying X contributes to the study of intelligence in general", where X can be replaced by reasoning, learning, planning, perceiving, acting, etc. On the contrary, we believe that most of the current AI research works make little direct contribution to AGI, though these works have value for many other reasons. Previously we have mentioned "machine learning" as an example. One of us (Goertzel) has published extensively about applications of machine learning algorithms to bioinformatics. This is a valid, and highly important sort of research – but it doesn't have much to do with achieving general intelligence.

There is no reason to believe that "intelligence" is simply a toolbox, containing mostly unconnected tools. Since the current AI "tools" have been built according to very different theoretical considerations, to implement them as modules in a big system will not necessarily make them work together, correctly and efficiently. Past attempts in this direction have taught us that "Component development is crucial; connecting the components is more crucial" [22].

Though it is possible to build AGI via an integrative approach, such integration needs to be guided by overall considerations about the system as a whole. We cannot blindly work on "parts", with the hope that they will end up working together. Because of these considerations, we think it is necessary to explicitly identify what we call "AGI" as different from mainstream AI research. Of course, even an AGI system still needs to be built step by step, and when the details of the systems are under consideration, AGI does need to use many results from previous AI research. But this does not mean that AGI reduces to an application of specialized AI components.

3.5. "It is too early to work on AGI"

Though many people agree that AGI is indeed the ultimate goal of AI research, they think it is premature to directly work on such a project, for various reasons.

For example, some people may suggest that AGI becomes feasible only after the research results regarding individual cognitive facilities (reasoning, learning, planning, etc) become mature enough to be integrated. However, as we argued above, without the guidance of an overall plan, these "parts" may never be ready to be organized into a whole.

A similar opinion is that the design of a general-purpose system should come out of the common features of various domain-specific systems, and therefore AGI can only be obtained by generalizing the design of many expert systems. The history of AI has not provided much support for this belief, which misses the point we make previously, that is, a general-purpose system and a special-purpose system are usually designed with very different assumptions, restrictions, and target problems.

Some people claim that truly intelligent systems will mainly be the product of more research results in brain science or innovations of hardware design. Though we have no doubt that the progress in these fields will provide us with important inspirations and tools, we do not see them as where the major AGI problems are. Few people believe that detailed emulation of brain structures and functions is the optimal path to AGI. Emulating the human brain in detail will almost surely one day be possible, but this will likely require massively more hardware than achieving an equivalent level of intelligence via other mechanisms (since contemporary computer hardware is poorly suited to emulating neural wetware), and will almost surely not provide optimal intelligence given the computational resources available. And, though faster and larger hardware is always desired, it is the AI researchers' duty to tell hardware researchers what kind of hardware is needed for AGI.

All the above objections to AGI have the common root of seeing the solution of AGI as depending on the solution of another problem. We haven't seen convincing evidence for this. Instead, AGI is more likely to be a problem that demands direct research, which it is not too early to start --- actually we think it is already pretty late to give the problem the attention it deserves.

3.6. "AGI is nothing but hype"

OK, let us admit it: AI got a bad name from unrealized predictions in its earlier years, and we are still paying for it. To make things worse, from time to time we hear claims, usually on the Internet, about "breakthrough" in AI research, which turn out to be completely groundless. Furthermore, within the research community, there is little consensus even on the most basic problems, such as what intelligence is and what the criteria are for research success in the field. Just see the example of Deep Blue: while some AI researchers take it as a milestone, some others reject it as mostly irrelevant to AI research. As a common effect of these factors, explicitly working on AGI immediately marks a researcher a possible crackpot.

As long as AGI has not been proved impossible, it remains a legitimate research topic. Given the well-known complexity of the problem, there is no reason to expect an AGI to reach its goal within a short period, and all the popular theoretical controversies will probably continue to exist even after an AGI has been completed as planned. The fact that there is little consensus in the field should make us more careful when judging a new idea as completely wrong. As has happened more than once in the history of science, a real breakthrough may come from a counter-intuitive idea.

On the other hand, the difficulty of the problem cannot be used as an excuse for loose discipline in research. Actually, in the AGI research field we have seen works that are as serious and rigorous as scientific results in any other area. Though the conceptions and techniques of almost all AGI projects may remain controversial in the near future, this does not mean that the field should be discredited – but rather that attention should be paid to resolving the outstanding issues through concerted research.

3.7. *"AGI research is not fruitful"*

Some oppositions to AGI research come mainly from practical considerations. Given the nature of the problem, research results in AGI are more difficult to obtain, more difficult to get accepted by the research community even once obtained, and more difficult to turn into practical products even once accepted. Compared to other fields currently going under the AI label, researchers in AGI are less likely to be rewarded, in terms of publication, funding, career opportunity, and so on.

These issues are all true, as the experience of many AGI researchers shows. Because of this, and also because AGI does not invalidate the other research goals currently in vogue in the AI community (as discussed previously), we are not suggesting the whole AI community to turn to AGI research. Instead, we only hope AGI research to get the recognition, attention, and respect it deserves, as an active, productive and critical aspect of the AI enterprise. Given the potential importance of this topic, such a hope should not be considered as unrealistic.

3.8. *"AGI is dangerous"*

This is another objection that is as old as the field of AI. Like any science and technology, AGI has the danger of being misused, but this is not a reason to stop AGI research, just as it is not a reason to stop scientific research in many other fields. The viewpoint that "AGI is fundamentally dangerous because it will inevitably lead to disaster" is usually based on various misconceptions about intelligence and AGI. For example, some version of this claim is based on the assumption that an intelligent system will eventually want to dominate the universe, which has no scientific evidence.

Like scientists and engineers in any domain, AGI researchers should be responsible for the social impacts of their work. Given the available evidence, we believe AGI research has a much larger chance to have benign consequences to the human beings than harmful ones. Therefore, we do not think AGI research should be stopped because of its possible danger, though we do agree that it is an issue that should be kept in the mind of every AGI researcher.

4. Building an AGI community

As discussed above, based on extrapolating recent trends, it can reasonably be anticipated that the AGI field will soon end its decades-long dormancy, and enter a period of awakening. Though each individual AGI approach still has many obstacles to overcome, more and more people will appreciate the value of this sort of research.

In the AGIRI workshop, a topic that was raised by many attendances is the need to develop an AGI research community. From direct personal experience, many AGI researchers strongly feel that the existing platforms of conferences and societies, as well as the channels of publication and funding, do not properly satisfy their needs for communication, coordination, cooperation, and support. As we argued above, AGI has its own issues, which have been mostly ignored by the mainstream AI community. AGI researchers have been working mostly in isolation, and finding themselves surrounded by researchers with very different research interests and agenda. As commented by an attendance of the AGIRI workshop, "I don't think in my long career (I'm getting quite old) I've ever been to a conference or workshop where I want to listen to such a large percentage of talks, and to meet so many people."

Though communication with other AI researchers is still necessary, the crucial need of the AGI field, at the current time, is to set up the infrastructures to support the regular communication and cooperation among AGI researchers. In this process, a common language will be developed, the similarities and differences among approaches will be clarified, repeated expenses will be reduced, and evaluation criteria will be formed and applied. All these are required for the growth of any scientific discipline.

Several community-forming activities are in the planning phase, and if all goes well, will be carried out soon. Their successes require the support of all AGI researchers, who will benefit from them in the long run.

5. This collection

The chapters in this book have been written by some of the speakers at the AGIRI Workshop after the meeting; each of them is based on a workshop talk, and also takes into account the feedback and afterthought of the meeting, as well as relationships with previous publications. Rather than thoroughly summarizing the contents of the chapters, here we will briefly review each chapter with a view toward highlighting its relationships with other chapters, so as to give a feeling for how the various approaches to and perspective on AGI connect together, in some ways forming parts of an emerging unified understanding in spite of the diversity of understanding perspectives.

First of all, following this chapter, Legg and Hutter's chapter (the only chapter whose authors did not attend the AGIRI Workshop) contains a simple enumeration of all the scientifically serious, published definitions of "intelligence" that the authors could dig up given a reasonable amount of effort. This is a worthwhile exercise in terms of illustrating both the commonality and the divergence among the various definitions. Clearly, almost all the authors cited are getting at similar intuitive concept – yet there are many, many ways to specify and operationalize this concept. And, of course, the choice of a definition of intelligence may have serious implications regarding one's preferred research direction. For instance, consider two of the definitions they cite:

> "Intelligence measures an agent's ability to achieve goals in a wide range of environments." -- S. Legg and M. Hutter

> "Intelligence is the ability for an information processing system to adapt to its environment with insufficient knowledge and resources." -- P. Wang

Note that the latter refers to limitations in processing power, whereas the former does not. Not surprisingly, much of the research of the authors of the prior definition concerns the theory of AGI algorithms requiring either infinite or extremely large amounts of processing power; whereas the central research programme of the latter author aims at achieving reasonable results using highly limited computational power.

Given this interconnectedness between the specifics of the definition of intelligence chosen by a researcher, and the focus of the research pursued by the researcher, it seems best to us that, at this stage of AGI research, the definition of intelligence be left somewhat loose and heterogeneous in the field, so as to encourage a diversity of conceptual approaches to the AGI problem. A loose analogy, in another field, might be the definition of "life" in biology. There is a clear intuitive meaning to "life," yet pinning down exactly what the term means has proven difficult – and has not really proved necessary for the progress of the field of biology. Rather, different interpretations regarding the essential nature of "life" have led to different, fruitful scientific developments; and, of course, the vast majority of research in areas as divergent as systems biology and genomics has progressed without much attention to the definitional issue. Of course, we are not suggesting that all definitions of intelligence are equally valid, or that different definitions cannot be compared – on the contrary, to identify the research goal is often the key to understand an AGI project, as discussed previously.

The next paper, "A Foundational Architecture for General Intelligence" by Stan Franklin, serves (at least) two purposes. This paper corresponds to the talk that opened up the workshop, and serves both to introduce Franklin's LIDA architecture for AGI, and also to propose a general framework for discussing and comparing various AGI systems. This latter purpose is taken up in the following chapter, entitled "Four Contemporary AGI Designs: A Comparative Treatment," in which four individuals who presented at the workshop and contributed chapters to this volume present answers to a series of 15 questions regarding their AGI architectures. These questions were mainly drawn from Franklin's article, and represent an attempt to take a first step toward a common framework for comparing different approaches to AGI.

One of the main contributions of Franklin's chapter is to systematically map connections between current understanding of the human mind, as reflected in the cognitive science literature, and the components of an AGI design. This has been done before, but Franklin does a particularly succinct and lucid job, and for those of us who have been following the field for a while, it is pleasing to see how much easier this job gets as time goes on, due to ongoing advances in cognitive science as well as AGI. Another interesting point is how similar the basic high-level "boxes and lines architecture diagrams" for various AGI architectures come out to be. Of course there is nothing like a universal agreement, but it seems fair to say that there is a rough and approximate agreement among a nontrivial percentage of contemporary AGI researchers regarding the general way that cognitive function may be divided up into sub-functions within an overall cognitive architecture. This fact is particularly interesting to the extent that it allows attention to focus less on the cognitive architecture than on the "pesky little details" of what happens inside the boxes and what passes along the lines between the boxes (of course, the phrase "pesky little details" is chosen with some irony, since many researchers believe that it is these details of learning and knowledge representation, rather than the overall cognitive architecture, that most deserve the label of the "essence of intelligence").

The following paper, by Eric Baum, seeks to focus in on this essence. Rather than giving an overall AGI architecture, Baum concentrates on what he sees as the key issue facing those who would build AGI: the "inductive bias" that he believes human brains derive from their genetic heritage. Baum's hypothesis is that the problem of learning to act like an ordinary human is too hard to be achieved by general-purpose learning algorithms of the quality embodied in the brain. Rather, he suggests, much of learning to act like a human is done via specialized learning algorithms that are tuned for the specific learning problems, such as recognizing humans face; or by means of specialized data that is fed into general learning algorithms, representing problem-specific bias. If this hypothesis is correct, then AGI designers have a big problem: even if they get the cognitive architecture diagram right, and plug reasonably powerful algorithms into the boxes carrying out learning, memory, perception and so forth, then even so, the algorithms may not be able to carry out the needed learning, because of the lack of appropriate inductive biases to guide them on their way.

In his book *What Is Thought?* [2], this problem is highlighted but no concrete solution is proposed. In his chapter here, Baum proposes what he sees as the sketch of a possible solution. Namely, he suggests, we can explicitly program a number of small "code modules" corresponding to the inductive bias supplied by the genome. AGI learning is then viewed as consisting of learning relatively simple programs that combine these code modules in task-appropriate ways. As an example of how this kind of approach may play out in practice, he considers the problem of writing a program that learns to play the game of Sokoban, via learning appropriate programs combining core modules dealing with issues like path-finding and understanding spatial relationships.

The next chapter, by Pei Wang (one of the authors of this Introduction), reviews his AGI project called NARS (Non-Axiomatic Reasoning System) which involves both a novel formal and conceptual foundation, and a software implementation embedding many aspects of the foundational theory. NARS posits that adaptation under knowledge-resources restriction is the basic principle of intelligence, and uses an AGI architecture with an uncertain inference engine at its core and other faculties like language processing, perception and action at the periphery, making use of specialized code together with uncertain inference. Compared to Franklin's proposed AGI architecture, NARS does not propose a modular high-level architecture for the core system, but places the emphasis on an uncertain inference engine implementing the proper semantics of uncertain reasoning. Regarding Baum's hypothesis of the need to explicitly code numerous modules encoding domain-specific functionalities (considered as inductive biases), Wang's approach would not necessarily disagree, but would consider these modules as to be built by the AGI architecture, and so they do not constitute the essence of intelligence but rather constitute learned special-purpose methods by which the system may interface with the world. Since the details of NARS have been covered by other publications, such as [17], this chapter mainly focuses on the development plan of NARS, which is a common issue faced by every AGI project. Since NARS is an attempt to minimize AGI design, some functionalities included in other AGI designs are treated as optional in NARS.

Nick Cassimatis's chapter presents a more recently developed approach to AGI architecture, which focuses on the combination of different reasoning and learning algorithms within a common framework, and the need for an integrative framework that can adaptively switch between and combine different algorithms depending on context. This approach is more similar to Franklin's than Wang's in its integrative nature, but differs from Franklin's in its focus on achieving superior algorithmic performance via hybridizing various algorithms, rather than interconnecting different algorithms in an overall architecture that assigns different algorithms strictly different functional roles. Cassimatis's prior work has been conceptually critical in terms of highlighting the power of AI learning algorithms to gain abstract knowledge spanning different domains – e.g. gaining knowledge about physical actions and using this knowledge to help learn language. This brings up a key difference between AGI work and typical, highly-specialized AI work. In ordinary contemporary AI work, computational language learning is one thing, and learning about physical objects and their interrelationships is something else entirely. In an integrated intelligent mind, however, language and physical reality are closely interrelated. AGI research, to be effective, must treat these interconnections in a concrete and pragmatic way, as Cassimatis has done in his research.

Alexei Samsonovich and Giorgio Ascoli, the authors of the next chapter, are also involved with developing an ambitious AGI architecture, called BICA-GMU, created with funding from DARPA. Their architecture has been described elsewhere, and bears a family resemblance to LIDA in that it uses a (very LIDA-like) high-level architecture diagram founded on cognitive science, and fills in the boxes with a variety of different algorithms. So far the focus with BICA-GMU has been on declarative rather than procedural knowledge and learning, and the focus of these authors' contribution to this volume is along these lines. The chapter is called "Cognitive Map Dimensions of the Human Value System Extracted from Natural Language," and it reports some fascinating experiments in statistical language processing, oriented toward discovering "natural conceptual categories" as clusters of words that naturally group together in terms of their contexts of occurrence in text. The categories found by the authors' automated learning method have an obvious intuitive naturalness to them, and essentially the same categories were found to emerge from analysis of text in two different languages. Of course, these results are preliminary and could particularly use validation via analysis of texts in non-Western languages; but they are nonetheless highly thought-provoking. One is reminded of Chomsky's finding of universal grammatical patterns underlying various languages, which gives rise to the question of whether these grammatical patterns are innate, evolved "inductive bias" or learned/self-organized patterns that characterize spontaneously emerging linguistic structures. Similarly, the findings in this chapter give rise to the question of whether these conceptual categories represent innate, evolved inductive bias, versus learned/self-organized patterns that spontaneously emerge in any humanly embodied mind attempting to understand itself and the world. This sort of question may of course be explored via ongoing experimentation with teaching AGI systems like BICA-GMU and some of the other AGI systems described in this book: one can experiment with such systems both with and without programmer-supplied innate conceptual categories, and see how the progress and nature of learning is affected. (More specifically: this kind of experimentation can be done only with AGI systems whose knowledge representation supports explicit importation of declarative knowledge, which is the case with most of the AGI designs proposed in this book, but is not obviously the case e.g. with neural net architectures such as the one suggested in the following chapter, by Hugo de Garis.

De Garis's chapter is somewhat different from the preceding ones, in that it doesn't propose a specific AGI architecture, but rather proposes a novel tool for building the components of AGI systems (or, as De Garis terms it, "brain building"). Although most of the authors in this book come from more of a cognitive/computer science perspective, another important and promising approach to AGI involves neural networks, computational models of brain activity at varying levels of granularity. In a sense this is the lowest-risk approach to producing AGI, since after all the human brain is the best example of an intelligent system that we know right now. So, there is some good common sense in approaching AGI by trying to emulate brain function. Now, there is also a major problem with this approach, which is that we don't currently understand human brain function very well. Some parts of the brain are understood better than others; for example, Jeff Hawkins' [16] AI architecture is closely modeled on the visual cortex, which is one of the best-understood parts of the human brain. At the current time, rather than focusing on constructing neural net AGI systems based on neuroscience knowledge, De Garis is focused on developing tools for constructing small neural networks that may serve as components of such AGI systems. Specifically he is focused on the problem of evolutionary learning of small neural networks: i.e., given a specification of what a neural net is supposed to do, he uses evolutionary learning to find a neural net doing that thing. The principal novelty of his approach is that this learning is conducted in hardware, on a reprogrammable chip (a field-programmable gate array), an approach that may provide vastly faster learning that is achievable through software-only methods. Preliminary results regarding this approach look promising.

Loosemore's paper considers the methodology of AGI research, and the way that this is affected by the possibility that all intelligent systems must be classified as complex systems. Loosemore takes a dim view of attempts to create AGI systems using the neat, formal approach of mathematics or the informal, bash-to-fit approach of engineering, claiming that both of these would be severely compromised if complexity is involved. Instead, he suggests an empirical science approach that offers a true marriage of cognitive science and AI. He advocates the creation of novel software tools enabling researchers to experiment with different sorts of complex intelligent systems, understanding the emergent structures and dynamics to which they give rise and subjecting our ideas about AI mechanisms to rigorous experimental tests, to see if they really do give rise to the expected global system performance.

The next paper, by Moshe Looks, harks back to De Garis et al's paper in its emphasis on evolutionary learning. Like De Garis et al, Looks is concerned with ways of making evolutionary learning much more efficient with a view toward enabling it to play a leading role in AGI – but the approach is completely different. Rather than innovating on the hardware side, Looks suggests a collection of fundamental algorithmic innovations, which ultimately constitute a proposal to replace evolutionary learning with a probabilistic-pattern-recognition based learning algorithm (MOSES = Meta-Optimizing Semantic Evolutionary Search) that grows a population of candidate problem solutions via repeatedly recognizing probabilistic patterns in good solutions and using these patterns to generate new ones. The key ideas underlying MOSES are motivated by cognitive science ideas, most centrally the notion of "adaptive representation building" – having the learning algorithm figure out the right problem representation as it goes along, as part of the learning process, rather than assuming a well-tuned representation right from the start. The MOSES algorithm was designed to function within the Novamente AGI architecture created by one of the authors of this Introduction (Goertzel) together with Looks and others (and discussed in other papers in this volume, to be mentioned below), but also to operate independently as a program learning solution. This chapter describes some results obtained using stand-alone MOSES on a standard test problem, the "artificial ant" problem. More powerful performance is hoped to be obtained by synthesizing MOSES with the PLN probabilistic reasoning engine, to be described in Ikle' et al's chapter (to be discussed below). But stand-alone MOSES in itself appears to be a dramatic improvement over standard evolutionary learning in solving many different types of problems, displaying fairly rapid learning on some problem classes that are effectively intractable for GA/GP. (This, of course, makes it interesting to speculate about what could be achievable by running MOSES on the reconfigurable hardware FPGA discussed in De Garis's chapter. Both MOSES and FPGA's can massively speed up evolutionary learning – the former in a fundamental order-of-complexity sense on certain problem classes, and the latter by a large constant multiplier in a less problem-class-dependent way. The combination of the two could be extremely powerful.)

The chapter by Matthew Ikle' et al, reviews aspects of Probabilistic Logic Networks (PLN) -- an AI problem-solving approach that, like MOSES, has been created with a view toward integration into the Novamente AI framework, as well as toward stand-alone performance. PLN is a probabilistic logic framework that combines probability theory, term logic and predicate logic with various heuristics in order to provide comprehensive forward and backward chaining inference in contexts ranging from mathematical theorem-proving to perceptual pattern-recognition, and speculative inductive and abductive inference. The specific topic of this chapter is the management of "weight of evidence" within PLN. Like NARS mentioned above and Peter Walley's imprecise probability theory [23], PLN quantifies truth values using a minimum of two numbers (rather than, for instance, a single number representing a probability or fuzzy membership value). One approach within PLN is to use two numbers (s,n), where s represents a probability value, and n represents a "weight of evidence" defining how much evidence underlies that probability value. Another, equivalent approach within PLN is to use two numbers (L,U), representing an interval probability, interpreted to refer to a family of probability distributions the set of whose means has (L,U) as a b% confidence interval. The chapter discusses the relationship between these representations, and the way that these two-number probabilities may be propagated through inference rules like deduction, induction, abduction and revision. It is perhaps worth noting that PLN originally emerged, in 1999-2000, as an attempt to create a probabilistic variant of the NARS uncertain logic, although it has long since diverged from these roots. Part of the underlying motivation for both NARS and PLN is the assumption that AGIs must be able to carry out a diversity of inferences involving uncertain knowledge and uncertain conclusions, and thus must possess a reasonably robust method of managing all this uncertainty. Humans are famously poor at probability estimation [24];[25], but nonetheless we are reasonably good at uncertainty management in many contexts, and both PLN and NARS (but using quite different methods) attempts to capture this kind of pragmatic uncertainty management that humans are good at. A difference between the two approaches is that PLN is founded on probability theory and attempts to harmonize human-style robust uncertainty management with precise probabilistic calculations – using the notion that the former is appropriate when data is sparse, and gradually merges into the latter as more data becomes available. On the other hand, in NARS the representation, interpretation, and processing of uncertainty do not follow probability theory in general, though agree with it on special cases. Furthermore, precise probabilistic inference would be implemented as a special collection of rules running on top of the underlying NARS inference engine in roughly the same manner that programs may run on top of an operating system.

Following up on the uncertain-logic theme, the next chapter by Stephan Vladimir Bugaj and Ben Goertzel moves this theme into the domain of developmental psychology. Piaget's ideas have been questioned by modern experimental developmental psychology, yet remain the most coherent existing conceptual framework for studying human cognitive development. It turns out to be possible to create a Piaget-like theory of stages of cognitive development that is specifically appropriate to uncertain reasoning systems like PLN, in which successive stages involve progressively sophisticated inference control: simple heuristic control (the infantile stage); inductive, history-based control (the concrete operational stage); inference-based inference control (the formal stage); and inference-based modification of inference rules (the post-formal stage). The pragmatic implications of this view of cognitive development are discussed in the context of classic Piagetan learning problems such as learning object permanence, conservation laws, and theory of mind.

In a general sense, quite apart from the specifics of the developmental theory given in Goertzel and Bugaj's chapter, one may argue that the logic of cognitive development is a critical aspect of AGI that has received far too little attention. Designing and building AGI's is important, but once they are built, one must teach them and guide their development, and the logic of this development may not be identical or even very similar to that of human infants and children. Different sorts of AGIs may follow different sorts of developmental logic. This chapter discusses cognitive development specifically in the context of uncertain logic based AGI systems, and comparable treatments could potentially be given for different sorts of AGI designs.

The following chapter, by Ben Goertzel (one of the authors of this Introduction), discusses certain aspects of the Novamente AGI design. A comprehensive overview of the Novamente system is not given here, as several published overviews of Novamente already exist, but the highlights are touched and some aspects of Novamente that have not been discussed before in publications are reviewed in detail (principally, economic attention allocation and action selection). Commonalities between Novamente and Franklin's LIDA architecture are pointed out, especially in the area of real-time action selection. Focus is laid on the way the various aspects of the Novamente architecture are intended to work together to lead to the emergence of complex cognitive structures such as the self and the "moving bubble of attention." These ideas are explored in depth in the context of a test scenario called "iterated Easter Egg Hunt," which has not yet been experimented with, but is tentatively planned for the Novamente project in mid-2007. This scenario appears to provide an ideal avenue for experimentation with integrated cognition and the emergence of self and adaptive attention, and is currently being implemented in the AGISim 3D simulation world, in which the Novamente system controls a humanoid agent.

Novamente is an integrative architecture, in the sense that it combines a number of different learning algorithms in a highly specific way. Probabilistic logic is used as a common language binding together the various learning algorithms involved. Two prior chapters (by Looks, and Ikle' et al) reviewed specific AI learning techniques that lie at the center of Novamente's cognition (MOSES and PLN). The next two chapters discuss particular applications that have been carried out using Novamente, in each case via utilizing PLN in combination with other simpler Novamente cognitive processes.

The Heljakka et al chapter discusses the learning of some very simple behaviors for a simulated humanoid agent in the AGISim 3D simulation world, via a pure "embodied reinforcement learning" methodology. In Piagetan terms, these are "infantile-level" tasks, but to achieve them within the Novamente architecture nevertheless requires a fairly subtle integration of various cognitive processes. The chapter reviews in detail how perceptual pattern mining, PLN inference and predicate schematization (declarative-to-procedural knowledge conversion) have been used to help Novamente learn how to play the classic human-canine game of "fetch."

The last two chapters of the book are not research papers but rather edited transcriptions of dialogues that occurred at the workshop. The first of these, on the topic of the ethics of highly intelligent AGIs, was probably the liveliest and most entertaining portion of the workshop, highlighted by the spirited back-and-forth between Hugo de Garis and Eliezer Yudkowsky. The second of these was on the practicalities of actually creating powerful AGI software systems from the current batch of ideas and designs, and included a variety of "timing estimates" for the advent of human-level AGI from a number of leading researchers. These dialogues give a more human, less formal view of certain aspects of the current state of philosophical and pragmatic thinking about AGI by active AGI researchers.

All in all, it cannot be claimed that these chapters form a balanced survey of the current state of AGI research – there are definite biases, such as a bias towards symbolic and uncertain-reasoning-based systems versus neural net type systems, and a bias away from robotics (though there is some simulated robotics) and also away from highly abstract theoretical work a la Hutter [3] and Schmidhuber [26]. However, they do present a survey that is both broad and deep, and we hope that as a collection they will give you, the reader, a great deal to think about. While we have a long way to go to achieve AGI at the human level and beyond, we do believe that significant progress is being made in terms of resolving the crucial problem of AGI design, and that the chapters here do substantively reflect this progress.

References

[1] A. R. Jensen, The G Factor: the Science of Mental Ability, Psycoloquy: 10,#2, 1999
[2] E. Baum, What is Thought? MIT Press, 2004.
[3] M. Hutter, Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability, Springer, 2005.
[4] B. Goertzel. The Structure of Intelligence, Springer, 1993.
[5] B. Goertzel and C. Pennachin (editors), Artificial General Intelligence, Springer, 2007.
[6] B. Goertzel. The Hidden Pattern, BrownWalker, 2006.
[7] J. Searle, Minds, Brains, and Programs, Behavioral and Brain Sciences 3 (1980), 417-424.
[8] J. McCarthy, The Future of AI — A Manifesto, AI Magazine, 26(2005), Winter, 39
[9] R. Brachman, Getting Back to "The Very Idea", AI Magazine, 26(2005), Winter, 48–50
[10] P. Langley, Cognitive Architectures and General Intelligent Systems, AI Magazine 27(2006), Summer, 33-44.
[11] A. M. Turing, Computing machinery and intelligence, Mind LIX (1950), 433-460.
[12] A. Newell and H. A. Simon, GPS, a program that simulates human thought, E. A. Feigenbaum and J. Feldman (editors), Computers and Thought, 279-293, McGraw-Hill, 1963.
[13] A. Newell, Unified Theories of Cognition, Harvard University Press, 1990.
[14] D. G. Stork, Scientist on the Set: An Interview with Marvin Minsky, D. G. Stork (editor), HAL's Legacy: 2001's Computer as Dream and Reality, 15-30, MIT Press, 1997.
[15] N. J. Nilsson, Human-Level Artificial Intelligence? Be Serious! AI Magazine, 26(2005), Winter, 68–75.
[16] J. Hawkins and S. Blakeslee, On Intelligence, Times Books, 2004.
[17] P. Wang, Rigid Flexibility: The Logic of Intelligence, Springer, 2006.
[18] J. R. Lucas, Minds, Machines and Gödel, Philosophy XXXVI (1961), 112-127.
[19] H.L. Dreyfus, What Computers Still Can't Do, MIT Press, 1992.
[20] R. Penrose, The Emperor's New Mind: Concerning Computers, Minds, and the Laws of Physics, Oxford University Press, 1989.
[21] D. Chalmers, Contemporary Philosophy of Mind: An Annotated Bibliography, Part 4: Philosophy of Artificial Intelligence, http://consc.net/biblio/4.html
[22] A. Roland and P. Shiman, Strategic computing: DARPA and the quest for machine intelligence, 1983-1993, MIT Press, 2002.
[23] P. Walley: Towards a unified theory of imprecise probability. Int. J. Approx. Reasoning 24(2-3): 125-148 (2000)
[24] D. Kahneman, P. Slovic, & A. Tversky (editors), Judgment under Uncertainty: Heuristics and Biases. Cambridge, UK: Cambridge University Press, 1982.
[25] T. Gilovich,, D. Griffin & D. Kahneman (editors), Heuristics and biases: The psychology of intuitive judgment. Cambridge, UK: Cambridge University Press, 2002.
[26] J. Schmidhuber, Goedel machines: self-referential universal problem solvers making provably optimal self-improvements. In B. Goertzel and C. Pennachin (editors), Artificial General Intelligence, 2006.

Advances in Artificial General Intelligence: Concepts, Architectures and Algorithms
B. Goertzel and P. Wang (Eds.)
IOS Press, 2007
© 2007 The authors and IOS Press. All rights reserved.

A Collection of Definitions of Intelligence

Shane LEGG[a] and Marcus HUTTER[b]
[a]IDSIA, Galleria 2, Manno-Lugano CH-6928, Switzerland
shane@idsia.ch www.idsia.ch/~shane
[b]RSISE/ANU/NICTA, Canberra, ACT, 0200, Australia
marcus@hutter1.net www.hutter1.net

Introduction

“Viewed narrowly, there seem to be almost as many definitions of intelligence as there were experts asked to define it." R. J. Sternberg quoted in [1]

Despite a long history of research and debate, there is still no standard definition of intelligence. This has lead some to believe that intelligence may be approximately described, but cannot be fully defined. We believe that this degree of pessimism is too strong. Although there is no single standard definition, if one surveys the many definitions that have been proposed, strong similarities between many of the definitions quickly become obvious. In many cases different definitions, suitably interpreted, actually say the same thing but in different words. This observation lead us to believe that a single general and encompassing definition for arbitrary systems was possible. Indeed we have constructed a formal definition of intelligence, called universal intelligence [2], which has strong connections to the theory of optimal learning agents [3].

Rather than exploring very general formal definitions of intelligence, here we will instead take the opportunity to present the many informal definitions that we have collected over the years. Naturally, compiling a complete list would be impossible as many definitions of intelligence are buried deep inside articles and books. Nevertheless, the 70 odd definitions presented below are, to the best of our knowledge, the largest and most well referenced collection there is. We continue to add to this collect as we discover further definitions, and keep the most up to date version of the collection available online [4]. If you know of additional definitions that we could add, please send us an email.

Collective definitions

In this section we present definitions that have been proposed by groups or organisations. In many cases definitions of intelligence given in encyclopedias have been either contributed by an individual psychologist or quote an earlier definition given by a psychologist. In these cases we have chosen to attribute the quote to the psychologist, and have placed it in the next section. In this section we only list those definitions that either cannot be attributed to specific individuals, or represent a collective definition agreed upon by many individuals. As many dictionaries source their definitions from other dictionaries, we have endeavoured to always list the original source.

1.“The ability to use memory, knowledge, experience, understanding, reasoning, imagination and judgement in order to solve problems and adapt to new situations.” AllWords Dictionary, 2006

2.“The capacity to acquire and apply knowledge.” The American Heritage Dictionary, fourth edition, 2000

3.“Individuals differ from one another in their ability to understand complex ideas, to adapt effectively to the environment, to learn from experience, to engage in various forms of reasoning, to overcome obstacles by taking thought.” American Psychological Association [5]

4.“The ability to learn, understand and make judgments or have opinions that are based on reason” Cambridge Advance Learner's Dictionary, 2006

5.“Intelligence is a very general mental capability that, among other things, involves the ability to reason, plan, solve problems, think abstractly, comprehend complex ideas, learn quickly and learn from experience.” Common statement with 52 expert signatories [6]

6.“The ability to learn facts and skills and apply them, especially when this ability is highly developed.” Encarta World English Dictionary, 2006

7.“...ability to adapt effectively to the environment, either by making a change in oneself or by changing the environment or finding a new one ...intelligence is not a single mental process, but rather a combination of many mental processes directed toward effective adaptation to the environment.” Encyclopedia Britannica, 2006

8.“the general mental ability involved in calculating, reasoning, perceiving relationships and analogies, learning quickly, storing and retrieving information, using language fluently, classifying, generalizing, and adjusting to new situations.” Columbia Encyclopedia, sixth edition, 2006

9.“Capacity for learning, reasoning, understanding, and similar forms of mental activity; aptitude in grasping truths, relationships, facts, meanings, etc.” Random House Unabridged Dictionary, 2006

10.“The ability to learn, understand, and think about things.” Longman Dictionary or Contemporary English, 2006

11.“: the ability to learn or understand or to deal with new or trying situations: ... the skilled use of reason (2) :the ability to apply knowledge to manipulate one's environment or to think abstractly as measured by objective criteria (as tests)” Merriam-Webster Online Dictionary, 2006

12.“The ability to acquire and apply knowledge and skills.” Compact Oxford English Dictionary, 2006

13.“...the ability to adapt to the environment.” World Book Encyclopedia, 2006

14.“Intelligence is a property of mind that encompasses many related mental abilities, such as the capacities to reason, plan, solve problems, think abstractly, comprehend ideas and language, and learn.” Wikipedia, 4 October, 2006

15.“Capacity of mind, especially to understand principles, truths, facts or meanings, acquire knowledge, and apply it to practise; the ability to learn and comprehend.” Wiktionary, 4 October, 2006

16.“The ability to learn and understand or to deal with problems.” Word Central Student Dictionary, 2006

17."The ability to comprehend; to understand and profit from experience." Wordnet 2.1, 2006

18."The capacity to learn, reason, and understand." Wordsmyth Dictionary, 2006

Psychologist definitions

This section contains definitions from psychologists. In some cases we have not yet managed to locate the exact reference and would appreciate any help in doing so.

1. "Intelligence is not a single, unitary ability, but rather a composite of several functions. The term denotes that combination of abilities required for survival and advancement within a particular culture." A. Anastasi [7]

2."...that facet of mind underlying our capacity to think, to solve novel problems, to reason and to have knowledge of the world." M. Anderson [8]

3."It seems to us that in intelligence there is a fundamental faculty, the alteration or the lack of which, is of the utmost importance for practical life. This faculty is judgement, otherwise called good sense, practical sense, initiative, the faculty of adapting ones self to circumstances." A. Binet [9]

4."We shall use the term `intelligence' to mean the ability of an organism to solve new problems ..." W. V. Bingham [10]

5."Intelligence is what is measured by intelligence tests." E. Boring [11]

6."...a quality that is intellectual and not emotional or moral: in measuring it we try to rule out the effects of the child's zeal, interest, industry, and the like. Secondly, it denotes a general capacity, a capacity that enters into everything the child says or does or thinks; any want of 'intelligence' will therefore be revealed to some degree in almost all that he attempts;" C. L. Burt [12]

7."A person possesses intelligence insofar as he has learned, or can learn, to adjust himself to his environment." S. S. Colvin quoted in [13]

8."...the ability to plan and structure one's behavior with an end in view." J. P. Das

9."The capacity to learn or to profit by experience." W. F. Dearborn quoted in [13]

10."...in its lowest terms intelligence is present where the individual animal, or human being, is aware, however dimly, of the relevance of his behaviour to an objective. Many definitions of what is indefinable have been attempted by psychologists, of which the least unsatisfactory are 1. the capacity to meet novel situations, or to learn to do so, by new adaptive responses and 2. the ability to perform tests or tasks, involving the grasping of relationships, the degree of intelligence being proportional to the complexity, or the abstractness, or both, of the relationship." J. Drever [14]

11."Intelligence A: the biological substrate of mental ability, the brains' neuroanatomy and physiology; Intelligence B: the manifestation of intelligence A, and everything that influences its expression in real life behavior; Intelligence C: the level of performance on psychometric tests of cognitive ability." H. J. Eysenck.

12."Sensory capacity, capacity for perceptual recognition, quickness, range or flexibility or association, facility and imagination, span of attention, quickness or alertness in response." F. N. Freeman quoted in [13]

13.“...adjustment or adaptation of the individual to his total environment, or limited aspects thereof ...the capacity to reorganize one's behavior patterns so as to act more effectively and more appropriately in novel situations ...the ability to learn ...the extent to which a person is educable ...the ability to carry on abstract thinking ...the effective use of concepts and symbols in dealing with a problem to be solved ...” W. Freeman

14.“An intelligence is the ability to solve problems, or to create products, that are valued within one or more cultural settings.” H. Gardner [15]

15.“...performing an operation on a specific type of content to produce a particular product.” J. P. Guilford

16.“Sensation, perception, association, memory, imagination, discrimination, judgement and reasoning.” N. E. Haggerty quoted in [13]

17.“The capacity for knowledge, and knowledge possessed.” V. A. C. Henmon [16]

18.“...cognitive ability.” R. J. Herrnstein and C. Murray [17]

19.“...the resultant of the process of acquiring, storing in memory, retrieving, combining, comparing, and using in new contexts information and conceptual skills.” Humphreys

20.“Intelligence is the ability to learn, exercise judgment, and be imaginative.” J. Huarte

21.“Intelligence is a general factor that runs through all types of performance.” A. Jensen

22.“Intelligence is assimilation to the extent that it incorporates all the given data of experience within its framework ...There can be no doubt either, that mental life is also accommodation to the environment. Assimilation can never be pure because by incorporating new elements into its earlier schemata the intelligence constantly modifies the latter in order to adjust them to new elements.” J. Piaget [18]

23.“Ability to adapt oneself adequately to relatively new situations in life.” R. Pinter quoted in [13]

24.“A biological mechanism by which the effects of a complexity of stimuli are brought together and given a somewhat unified effect in behavior.” J. Peterson quoted in [13]

25.“...certain set of cognitive capacities that enable an individual to adapt and thrive in any given environment they find themselves in, and those cognitive capacities include things like memory and retrieval, and problem solving and so forth. There's a cluster of cognitive abilities that lead to successful adaptation to a wide range of environments.” D. K. Simonton [19]

26.“Intelligence is part of the internal environment that shows through at the interface between person and external environment as a function of cognitive task demands.” R. E. Snow quoted in [20]

27.“...I prefer to refer to it as `successful intelligence.' And the reason is that the emphasis is on the use of your intelligence to achieve success in your life. So I define it as your skill in achieving whatever it is you want to attain in your life within your sociocultural context — meaning that people have different goals for themselves, and for some it's to get very good grades in school and to do well on tests, and for others it might be to become a very good basketball player or actress or musician.” R. J. Sternberg [21]

28.“...the ability to undertake activities that are characterized by (1) difficulty, (2) complexity, (3) abstractness, (4) economy, (5) adaptedness to goal, (6) social value, and (7) the emergence of originals, and to maintain such activities under conditions that demand a concentration of energy and a resistance to emotional forces.” Stoddard

29.“The ability to carry on abstract thinking.” L. M. Terman quoted in [13]

30.“Intelligence, considered as a mental trait, is the capacity to make impulses focal at their early, unfinished stage of formation. Intelligence is therefore the capacity for abstraction, which is an inhibitory process.” L. L. Thurstone [22]

31.“The capacity to inhibit an instinctive adjustment, the capacity to redefine the inhibited instinctive adjustment in the light of imaginally experienced trial and error, and the capacity to realise the modified instinctive adjustment in overt behavior to the advantage of the individual as a social animal.” L. L. Thurstone quoted in [13]

32.“A global concept that involves an individual's ability to act purposefully, think rationally, and deal effectively with the environment.” D. Wechsler [23]

33.“The capacity to acquire capacity.” H. Woodrow quoted in [13]

34.“...the term intelligence designates a complexly interrelated assemblage of functions, no one of which is completely or accurately known in man ...” R. M. Yerkes and A. W. Yerkes [24]

35.“...that faculty of mind by which order is perceived in a situation previously considered disordered.” R. W. Young quoted in [25]

AI researcher definitions

This section lists definitions from researchers in artificial intelligence.

1.“...the ability of a system to act appropriately in an uncertain environment, where appropriate action is that which increases the probability of success, and success is the achievement of behavioral subgoals that support the system's ultimate goal.” J. S. Albus [26]

2.“Any system ...that generates adaptive behviour to meet goals in a range of environments can be said to be intelligent.” D. Fogel [27]

3.“Achieving complex goals in complex environments.” B. Goertzel [28]

4.“Intelligent systems are expected to work, and work well, in many different environments. Their property of intelligence allows them to maximize the probability of success even if full knowledge of the situation is not available. Functioning of intelligent systems cannot be considered separately from the environment and the concrete situation including the goal.” R. R. Gudwin [29]

5.“[Performance intelligence is] the successful (i.e., goal-achieving) performance of the system in a complicated environment.” J. A. Horst [30]

6.“Intelligence is the ability to use optimally limited resources – including time – to achieve goals.” R. Kurzweil [25]

7.“Intelligence is the power to rapidly find an adequate solution in what appears *a priori* (to observers) to be an immense search space.” D. Lenat and E. Feigenbaum [31]

8.“Intelligence measures an agent's ability to achieve goals in a wide range of environments.” S. Legg and M. Hutter [2]

9.“...doing well at a broad range of tasks is an empirical definition of `intelligence' ” H. Masum [32]

10.“Intelligence is the computational part of the ability to achieve goals in the world. Varying kinds and degrees of intelligence occur in people, many animals and some machines.” J. McCarthy [33]

11.“...the ability to solve hard problems.” M. Minsky [34]

12.“Intelligence is the ability to process information properly in a complex environment. The criteria of properness are not predefined and hence not available beforehand. They are acquired as a result of the information processing.” H. Nakashima [35]

13.“...in any real situation behavior appropriate to the ends of the system and adaptive to the demands of the environment can occur, within some limits of speed and complexity.” A. Newell and H. A. Simon [36]

14.“[An intelligent agent does what] is appropriate for its circumstances and its goal, it is flexible to changing environments and changing goals, it learns from experience, and it makes appropriate choices given perceptual limitations and finite computation.” D. Poole [37]

15.“Intelligence means getting better over time.” Schank [38]

16.“Intelligence is the ability for an information processing system to adapt to its environment with insufficient knowledge and resources.” P. Wang [39]

17.“...the mental ability to sustain successful life.” K. Warwick quoted in [40]

18.“...the essential, domain-independent skills necessary for acquiring a wide range of domain-specific knowledge – the ability to learn anything. Achieving this with `artificial general intelligence' (AGI) requires a highly adaptive, general-purpose system that can autonomously acquire an extremely wide range of specific knowledge and skills and can improve its own cognitive ability through self-directed learning.” P. Voss [41]

Is a single definition possible?

In matters of definition, it is difficult to argue that there is an objective sense in which one definition could be considered to be the correct one. Nevertheless, some definitions are clearly more concise, precise and general than others. Furthermore, it is clear that many of the definitions listed above are strongly related to each other and share many common features. If we scan through the definitions pulling out commonly occurring features we find that intelligence is:

- A property that an individual agent has as it interacts with its environment or environments.
- Related to the agent's ability to succeed or profit with respect to some goal or objective.
- Depends on the agent's ability to adapt to different objectives and environments.

Putting these key attributes together produces the informal definition of intelligence that we have adopted,

> “Intelligence measures an agent's ability to achieve goals in a wide range of environments.” S. Legg and M. Hutter

Features such as the ability to learn and adapt, or to understand, are implicit in the above definition as these capacities enable an agent to succeed in a wide range of environments. For a more comprehensive explanation, along with a mathematical formalisation of the above definition, see [2] or our forthcoming journal paper.

References

[1] R. L. Gregory. *The Oxford Companion to the Mind*. Oxford University Press, Oxford, UK, 1998.

[2] S. Legg and M. Hutter. A formal measure of machine intelligence. In *Annual Machine Learning Conference of Belgium and The Netherlands (Benelearn'06)*, Ghent, 2006.

[3] M. Hutter. *Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability*. Springer, Berlin, 2005. 300 pages, http://www.idsia.ch/~marcus/ai/uaibook.htm.

[4] S Legg and M. Hutter. www.idsia.ch/~shane/intelligence.html. *A collection of definitions of intelligence*, 2006.

[5] U. Neisser, G. Boodoo, T. J. Bouchard, Jr., A. W. Boykin, N. Brody, S. J. Ceci, D. F. Halpern, J. C. Loehlin, R. Perloff, R. J. Sternberg, and S. Urbina. Intelligence: Knowns and unknowns. *American Psychologist*, 51(2):77–101, 96.

[6] L. S. Gottfredson. Mainstream science on intelligence: An editorial with 52 signatories, history, and bibliography. *Intelligence*, 24(1):13–23, 1997.

[7] A. Anastasi. What counselors should know about the use and interpretation of psychological tests. *Journal of Counseling and Development*, 70(5):610–615, 1992.

[8] M.. Anderson. Intelligence. *MS Encarta online encyclopedia*, 2006.

[9] A. Binet and T. Simon. Methodes nouvelles por le diagnostic du niveai intellectuel des anormaux. *L'Ann? Psychologique*, 11:191–244, 1905.

[10] W. V. Bingham. *Aptitudes and aptitude testing*. Harper & Brothers, New York, 1937.

[11] E. G. Boring. Intelligence as the tests test it. *New Republic*, 35:35–37, 1923.

[12] .L. Burt. *The causes and treatments of backwardness*. University of London press, 1957.

[13] R. J. Sternberg, editor. *Handbook of Intelligence*. Cambridge University Press, 2000.

[14] J. Drever. *A dictionary of psychology*. Penguin Books, Harmondsworth, 1952.

[15] H.Gardner. *Frames of Mind: Theory of multiple intelligences*. Fontana Press, 1993.

[16] V. A. C. Henmon. The measurement of intelligence. *School and Society*, 13:151–158, 1921.

[17] R. J. Herrnstein and C. Murray. *The Bell Curve: Intelligence and Class Structure in American Life*. Free Press, 1996.

[18] J. Piaget. *The psychology of intelligence*. Routledge, New York, 1963.

[19] D. K. Simonton. An interview with Dr. Simonton. In J. A. Plucker, editor, *Human intelligence: Historical influences, current controversies, teaching resources*. http://www.indiana.edu/~intell, 2003.

[20]] J. Slatter. *Assessment of children: Cognitive applications*. Jermone M. Satler Publisher Inc., San Diego, 4th edition, 2001.

[21] R. J. Sternberg. An interview with Dr. Sternberg. In J. A. Plucker, editor, *Human intelligence: Historical influences, current controversies, teaching resources*. http://www.indiana.edu/~intell, 2003.

[22] L. L. Thurstone. *The nature of intelligence*. Routledge, London, 1924.

[23] D. Wechsler. *The measurement and appraisal of adult intelligence*. Williams & Wilkinds, Baltimore, 4 edition, 1958.

[24] R. M. Yerkes and A. W. Yerkes. *The great apes: A study of anthropoid life*. Yale University Press, New Haven, 1929.

[25] R. Kurzweil. *The age of spiritual machines: When computers exceed human intelligence*. Penguin, 2000.

[26] J. S. Albus. Outline for a theory of intelligence. *IEEE Trans. Systems, Man and Cybernetics*, 21(3):473–509, 1991.

[27] D. B. Fogel. Review of computational intelligence: Imitating life. *Proc. of the IEEE*, 83(11), 1995.

[28] B. Goertzel. *The Hidden Pattern*. Brown Walker Press, 2006.

[29] R. R. Gudwin. Evaluating intelligence: A computational semiotics perspective. In *IEEE International conference on systems, man and cybernetics*, pages 2080–2085, Nashville, Tenessee, USA, 2000.

[30] R. R. Gudwin. Evaluating intelligence: A computational semiotics perspective. In *IEEE International conference on systems, man and cybernetics*, pages 2080–2085, Nashville, Tenessee, USA, 2000.

[30] J. Horst. A native intelligence metric for artificial systems. In *Performance Metrics for Intelligent Systems Workshop*, Gaithersburg, MD, USA, 2002.

[31] D. Lenat and E. Feigenbaum. On the thresholds of knowledge. *Artificial Intelligence*, 47:185–250, 1991.

[32] H. Masum, S. Christensen, and F. Oppacher. The Turing ratio: Metrics for open-ended tasks. In *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 973–980, New York, 2002. Morgan Kaufmann Publishers.

[33]] J. McCarthy. What is artificial intelligence? *www-formal.stanford.edu/jmc/whatisai/whatisai.html*, 2004.

[34] M. Minsky. *The Society of Mind*. Simon and Schuster, New York, 1985.

[35] H. Nakashima. AI as complex information processing. *Minds and machines*, 9:57–80, 1999.

[36] A. Newell and H. A. Simon. Computer science as empirical enquiry: Symbols and search. *Communications of the ACM 19*, 3:113–126, 1976.

[37] D. Poole, A. Mackworth, and R. Goebel. *Computational Intelligence: A logical approach*. Oxford University Press, New York, NY, USA, 1998.

[38] R. Schank. Where's the AI? *AI magazine*, 12(4):38–49, 1991.

[39] P. Wang. On the working definition of intelligence. Technical Report 94, Center for Research on Concepts and Cognition, Indiana University, 1995.

[40] A. Asohan. Leading humanity forward. *The Star*, October 14, 2003.

[41] P. Voss. Essentials of general intelligence: The direct path to AGI. In B. Goertzel and C. Pennachin, editors, *Artificial General Intelligence*. Springer-Verlag, 2005.

Advances in Artificial General Intelligence: Concepts, Architectures and Algorithms
B. Goertzel and P. Wang (Eds.)
IOS Press, 2007

© 2007 The authors and IOS Press. All rights reserved.

Four Contemporary AGI Designs: a Comparative Treatment

September 26, 2006
Stan FRANKLIN, Ben GOERTZEL, Alexei SAMSONOVICH, Pei WANG

Introduction (by Ben Goertzel)

During his talk at the AGI Workshop, Stan Franklin suggested that his LIDA architecture might fruitfully be considered not only as a specific AGI design, but also as a general framework within which to discuss and compare various AGI designs and approaches. With this in mind, following the workshop itself, I (Goertzel) formulated a list of simple questions intended to be pertinent to any AGI software design, mostly based on the conceptual framework presented in Stan Franklin's workshop presentation (and represented in this volume by his article "A Foundational Architecture for Artificial General Intelligence") with a couple additions and variations. All individuals who presented talks on AGI architectures at the workshop were invited to respond to the questionnaire, giving answers appropriate to their own AGI design. Four individuals (myself, Wang, Franklin and Samsonovich) took up this offer, and their answers are reported here – without modification; exactly as they gave them. All of these designs are described to some extent elsewhere in the book.

I find this style of reportage an interesting one, and pleasantly different from the standard approach in which each AGI system is described in its own paper in terms of its own specialized terminology, and with its own peculiar focus. Potentially, one could consider this simple questionnaire as a first step toward the creation of a systematic typology and ontology of AGI approaches. The question of what are the key questions to ask about an AGI system, if one wishes to understand how it differs from other AGI systems and what are its truly essential aspects, is an interesting question unto itself, and asking it leads one straight into the heart of the AGI enterprise. In future I would find it interesting to repeat this sort of experiment with a more detailed questionnaire and a larger group of AGI system designers.

1. Questions and Responses

- What kind of **environment** is your AI system intended to interact with?

Franklin:

The external environment of the original IDA consisted of email messages from sailors, and of responses to database queries. The later LIDA (Learning IDA) is being developed in two domains, one to control robots in a University environment (ROBLIDA), and the other in an image database environment (ILIDA).\

Goertzel:

Novamente may interact with any sort of environment, including totally nonphysical environments such as pools of mathematical theorems. However, we are currently utilizing it in two contexts. One is language processing, in which the environment is simply text that passes back and forth between it and human users or human document repositories. And the other is the control of a simple humanoid agent in a 3D simulation world called AGISim. Actual control of physical robots is also of interest to us, but we haven't gotten there yet, for pragmatic rather than principled reasons.

Samsonovich:

Currently, it works in a simplistic 2-D virtual environment. However, the architecture is designed to be able to handle any environmental embedding (real or virtual), provided the interface capabilities are implemented and available to the architecture.

Wang:

NARS interacts with its environment in real time. The environment can feed the system with any task (new knowledge, question, or goal) that is expressible in Narsese, the knowledge representation language of the system. No restriction is made on the content and timing of the tasks.

- What mechanisms are used for "**perception**" (in the sense of mapping sensory data into a set of relationships that can be used by the AI system to make significant practical decisions)?

Franklin:

IDA and LIDA model perception using ideas from the Copycat Architecture of Hofstadter and Mitchell.

Goertzel:

There is a special perception subsystem called the Perceptual Pattern Miner. Basically, it recognizes repeated spatiotemporal patterns in sensory inputs. The same algorithm is applied to perceptual inputs, to records of actions taken, and also to records of cognitions taken – so perception spans sensation, action and thought, rather than just being perception of the external world. Algorithmically, the Perceptual Pattern Miner in its currently implemented version searches for patterns that are conjunctions of predicates, where the predicates it looks for are provided to it from two sources: some hard-wired predicates related to the structure of space and time, and predicates flagged as important by cognition. The current version doesn't take the hierarchical structure of perceptual patterns directly into account but a future version may.

Samsonovich:

A set of special functions called "primitives" that constitute the procedural memory are used to do sensory perception. We assume that these functions will be able to do signal-to-symbol conversion in any given environment / embodiment.

Examples of top-down control: the architecture may send a request to the sensory system to pay attention to a particular location, to look for a particular feature, or to re-examine acquired data.

Wang:

Sensors are optional parts of NARS. The system can accept any sensor that can be invoked by an operation in the format of (*operator, argument-list*), and can produce effects expressible in Narsese.

- What mechanisms are used for the execution of procedures embodying **external actions**? (e.g. procedures involving coordinated sets of specific commands to be sent to actuators)?

Franklin:

In an essentially rule-based manner, IDA filled in the blanks in built-in scripts. ROBLIDA will likely use subsumption networks a la Brooks. I have no answer as yet for ILIDA.

Goertzel:

The Combo language, which represents procedures inside Novamente, contains primitives correcponding to specific external actions (such as, e.g.: "move this joint by this amount at this angle").

Samsonovich:

Again, special functions called "primitives" are used to execute actions. In particular, they are supposed to provide the higher symbolic level with a proprioceptive feedback.

Wang:

Effectors are optional parts of NARS. The system can accept any effector that can be invoked by an operation in the format of (*operator, argument-list*), and can produce effects expressible in Narsese.

- How are **procedures** stored in memory? (This has two aspects: what form do individual procedures take in memory; and how is the overall memory store of procedures organized?)

Franklin:

IDA used behavior codelets. LIDA uses a scheme net (schema a la Drescher). The links in the scheme net are "derived from" links.

Goertzel:

Executable procedures are stored in data objects which are equivalent to programs in a simple, LISP-like language called Combo. These Procedure objects are stored in a ProcedureRepository; and each of these objects is associated with a ProcedureNode that resides in semantic, declarative memory. Implicit knowledge about procedures is contained in declarative memory, and may be used to create new Procedure objects via a process called "predicate schematization." A key point is that Procedures, whenever possible, are broken down into small modules: this makes for effective real-time context-switching between various currently active procedures; and also makes procedure learning and procedural inference much easier.

Samsonovich:

Again, the lower-level (interface) procedures are stored as special function in a separate module called "procedural memory". Currently, there is no definite specification as to how the procedural memory should be organized. In any case, procedural memory is not the focus of our approach.

Overall, our architecture has four memory systems: procedural, working, semantic and episodic, plus the input-output buffer (in addition to the other 3 components; together it has 8 components).

Wang:

Procedures are represented as operations, which are statements under procedural interpretation. An atomic operation has the format of (*operator, argument-list*), and compound operations are formed from simpler operations and statements by logical operators.

- How are **episodes** (experienced by the system itself) stored in memory? (Again, this has two aspects: what form do individual episodes take in memory; and how is the overall memory store of episodes organized?)

Franklin:

Episodic memory is implemented by variants of Kanerva's sparse distributed memory. In IDA episodic structure was hand-crafted to fit the domain. In early experimentation for LIDA Philmore's case grammer was use. I don't yet know what will follow.

Goertzel:

Temporal relationships are represented using a special form of temporal logic, which extends Novamente's standard "probabilistic logic networks" variant of probabilistic logic, and integrates probabilistic knowledge representation with ideas from "event calculus." Episodes are then collections of knowledge bound together by temporal relationships -- and which form "maps" in Novamente's knowledge base, in the sense that an episode is a set of knowledge-items that the system frequently finds it useful to utilize as a whole.

Samsonovich:

Episodic memories are stored in our architecture as mental states. Each mental state is an instance of a Self of the agent taken together with all its current experiences (represented as instances of schemas).

Overall, episodic memory is clustered into episodes. A set of mental states may form a cluster (an episode) based on their logical and functional interdependence as well as proximity in time and space. An episode is comparable to a single snapshot of the entire working memory.

Wang:

Episodes are represented as events, which are statements with temporal attributes. Like all statements, they are stored in the related concepts.

- How are facts and conjectures ("**declarative knowledge**") stored in memory? (Again, this has two aspects: what form do individual facts/conjectures take in memory; and how is the overall memory store of facts/conjectures organized?)

Franklin:

Facts and conjectures are constructed in the workspace and are stored in declarative memory, perhaps eventually becoming part of it's subset, semantic memory.

Goertzel:

Declarative knowledge is represented using a semantic network type approach, involving types nodes and links. Some nodes represent abstract concepts, some represent particular objects, some represent percepts or actions, and some represent patterns not well described by any of these terms. Links, depending on type, may represent relationships such as logical inheritance, implication, equivalence, or else Hebbian-style association or ordered or unordered grouping, etc.

Finally, as well as knowledge stored explicitly in individual logical links, there is also implicit knowledge, stored in the pattern of linkages. For instance, if a set of nodes are tightly interlinked with Hebbian association links, then when a few nodes in the set are utilized, the others will tend to be utilized afterwards (due to the system's attention allocation dynamics) – so the set as a whole may be implicitly used to represent an item of knowledge.

Samsonovich:

Semantic memory in our architecture stores general knowledge of the agent. It consists of a set of schemas organized by a global semantic net (each schema corresponds to a node in this net). A schema is a very general form of representation that allows us to represent any given category or a class of categories. It can be conceived as a graph or as a 2-D table of nodes. All nodes have one and the same standard set of attributes (about 20 attributes). Nodes are connected to each other via bindings, etc.

In addition, semantic memory includes a part called "reference memory" which represents agent's beliefs about the most current actual state of the world. E.g., reference memory can be implemented as a spatial map with instances of schemas allocated on it. We still have debates regarding the implementation of reference memory.

Wang:

All declarative knowledge are represented as Narsese statements, and stored in the related concepts.

- How does "**attention**" work? How is it quantified and what mechanisms control its change over time? (Here attention is considered as the process that brings information from perception and long-term (episodic/declarative) memory into the attentional focus.)

Franklin:

Attention is the work of attention codelets who gather coalitions that compete for access to consciousness, the global workspace. The winner is broadcast throughout the cognitive system.

Goertzel:

Each node or link in Novamente is labeled with two numbers collectively comprising an "attention value," called the short-term and long-term importance. Roughly, the former governs processor time allocation and the latter governs whether the node/link is allowed to stay in memory or not. Importance levels are dynamically updated using a set of equations based on "artificial economics", involving two currencies, one for short term and one for long term importance. So, for instance, the "moving bubble of attention" is an emergent dynamical phenomenon, which at any point in time consists of the nodes/links with the highest short-term importance.

Samsonovich:

Each schema node has an attribute "attention" that determines the probability of selecting this node for processing, etc. Special rules implemented in our "driving engine" (one of the 8 components of the architecture) control the dynamics of attention.

Wang:

Attention works in NARS as a dynamic resources allocation mechanism. Each task has a priority indicating its share of processing time, and each belief has a priority indicating is accessibility.

- How does "**filtering**" work – i.e., how does the attentional mechanism cope with filtering the vast number of items that may be seeking attention at any given time?

Franklin:

Filtering occurs in the sensory mechanism, by the selection of the sensors. It occurs again during perception as the percept is formed. It occurs when local associations are cued from the episodic memories. It occurs during the competition of coalitions for consciousness. It occurs during instantiation of schemes from procedural memory. Finally, it occurs during the selection of a single action at the end of a cognitive cycle. Filtering can also occur during higher-level cognitive processing, say during deliberation or volition. Filtering seems to be ubiquitous during cognition.

Goertzel:

Filtering of sensory data occurs implicitly via the attention allocation mechanism (the dynamics of short-term importance levels). Low-level percepts are assigned a low-level of long-term importance, and in most cases never get a high level, so they're almost always forgotten shortly after they arise. The patterns recognized among these low-level percepts generally are assigned a higher level of long-term importance, and are more likely to be remembered.

Samsonovich:

There are several mechanisms that can be called "filtering mechanisms" in our architecture; most of them do not involve attention control. First, there are "syntactic" rules of binding schemas and their instances to each other. Then, filtering of candidates for binding is done with the help of our neuromorphic cognitive maps (one of the 8 components). Finally, there are special attributes (activation, attention) that in effect do filtering.

Wang:

Filtering works in NARS as the consequence of resources competition among tasks, beliefs, and concepts. Only the items with high priority get time-space resources needed for their processing.

- How does "**action selection**" work? For example, how is the choice made between two tasks serving two different concurrent goals?

Franklin:

Action selection employs an enhancement of Maes' behavior net. It deals effectively with concurrent goals, dangers, unexpected opportunities, etc.

Goertzel:

Action selection is carried out using an application of economic attention allocation, which may (taking some liberties) be thought of as an extension/modification of Maes' behavior net. Actions are represented by Procedure objects that are broken down into modules, and these modules are connected both by probabilistic logical implication relationships (representing "precondition" relationships between modules) and by activation relationships along which currency values representing "short term importance" pass between modules. The dynamics of action selection then emerge as a consequence of the dynamics of probabilistic reasoning and economic attention allocation. This dynamics can pass action back and forth between different procedures in the "active procedure pool" of currently active procedures, via the rule that, when multiple modules in the active procedure pool have their preconditions met but cannot be simultaneously executed without interfering with each other, then the one with the largest short-term-importance currency is selected for execution.

Samsonovich:

There is a special higher-level procedure of a voluntary action implemented as a part of our driving engine. (There is a set of higher-level procedures, including perception, understanding, voluntary action, checking the result, conflict resolution, and many more; however, these are not parts of procedural memory, as they are executed in full awareness of the system of what and how is done).

Wang:

Usually "selection" becomes "distribution", in the sense that all the tasks will be processed, but at different speeds. As a special case, new goals are created by a decision-making mechanism from desirable and plausible events.

- What **learning mechanisms** exist in your AI system? For each mechanism, specify: what kinds of learning problems is it specialized for, and how does it interact with other learning mechanisms?

Franklin:

IDA did not learn. LIDA implements perceptual, episodic, and procedural learning, and later will include attentional learning. They interact only indirectly.

Goertzel:

Very broadly speaking, there are three main learning mechanisms in Novamente: probabilistic inference, (probabilistic) evolutionary learning, and greedy stochastic pattern mining. Each of these can appeal to the others internally to help it out, quite directly. Furthermore, each of these can be used in multiple ways. For instance, probabilistic inference and evolutionary learning can both be used for both declarative and procedural learning; and, conjunctive pattern mining is used for both perceptual pattern mining and for "map formation" (recognition, and explicit representation, of concepts that exist only implicitly as collections of commonly-utilized nodes/links).

Samsonovich:

There is a number of learning mechanisms, ranging in complexity from almost trivial episodic memory creation (by transferring mental states from working memory to episodic memory) to new schema creation, off-line self-analysis and cognitive map self-organization (involved in, e.g., building a system of values). This is a long story.

Wang:

All object-level knowledge in NARS can be learned, by several mechanisms:

a) New tasks/beliefs/concepts can be accepted from the environment;
b) New tasks and beliefs can be derived from existing ones by inference rules;
c) The truth-value of beliefs can be modified by the revision rule;
d) New concepts can be formed from existing concepts by compound-term composition/decomposition rules;
e) The priority values of tasks/beliefs/concepts can be adjusted by the feedback evaluation mechanism.
f) Existing tasks/beliefs/concepts can be removed by the forgetting mechanism.

Since each of the above mechanisms works on a different aspect of the memory, it does not need to directly interact with the others.

- How is **meta-cognition** (reflective cognition about cognition, leading potentially to goal-directed self-modification of cognitive processes) implemented in your system?

Franklin:

We implemented two versions of meta-cognition in IDA, each as a Minsky style B-brain, and each using a different mechanism. This was the wrong way to approach the problem. In LIDA we'll implement meta-cognition as a collection of behavior streams that will affect the system internally over multiple cycles.

Goertzel:

In principle, there should be no strict distinction between meta-cognition and cognition in the Novamente system. All the system's cognitive processes may be represented as Procedure objects in the same format as procedures that the system learns; and ultimately, the system should be able to learn new cognitive processes, and modify its existing ones, using the same techniques it uses for general procedure learning. At the moment, however, for reasons of computational efficiency, many of the system's cognitive processes are represented as C++ code, which prevents them from being adaptively modified as the system grows and learns. It is intended to change this situation in future when a more efficient interpreter of learned (Combo) procedures is implemented.

At the moment, metacognition exists in the system in a more limited way, however. For instance, the heuristics used for pruning of inference processes involve utilization of patterns mined from prior inferences. And, the probabilistic evolutionary learning method used in Novamente (MOSES) looks for patterns across multiple attempts at solving the same problem, and even attempts at solving related problems, thus learning new representations of problem space.

Samsonovich:

The system has access to all its higher-level internal representations: mental states and their parts. Mental states can process other mental states, as they would process sensory information or intended behavioral actions. Some mental states do just this. For example, a mental state "I-Meta" makes sure that the rest of working memory form a consistent working scenario.

Wang:

Certain type of meta-cognition can be achieved as self-perception and self-control with certain internal sensors and effectors. More radical changes to the system design is left for an evolution process, which is separated from the intelligence mechanism that works within the life-cycle of a single system.

- How does **deliberation** occur in your AI system? (Deliberation is taken to refer to activities such as planning, deciding, scheduling, etc. that require "conscious thinking" about some issue.)

Franklin:

Deliberation was implemented in IDA, and will be in LIDA, as a higher-level process operating over multiple cycles by means of behavior streams.

Goertzel:

"Deliberation" in Novamente is directed by (forward or backward chaining) probabilistic inference, supported by other cognitive processes as it finds necessary.

Samsonovich:

As mentioned above, our architecture has a standard procedure for a voluntary action (and most actions that the agent does are voluntary). This procedure includes such elements as generation of ideas (i.e., instances f schemas representing feasible actions in the current situation), selection of those that fit best into the working scenario, making an intent, scheduling and performing its execution, checking results, and so on.

Wang:

Many forms of deliberation occur in NARS as reasoning on operations. In NARS, the conscious thinking and subconscious thinking are carried out by the same mechanism, and their difference is mainly quantitative, not qualitative.

- How does **creativity** occur in your AI system? Specifically, "creativity" is taken to refer to the creation of surprising new concepts or conjectures that were not obviously implicit in the knowledge given to the system.

Franklin:

There's little, if any, creativity in IDA. LIDA should be capable of creativity via perceptual learning, which is done on a generate-and-test basis. Attentional learning may also play a role.

Goertzel:

Creativity occurs in multiple ways, including within the probabilistic inference module (e.g. through inductive and abductive inference), evolutionary learning (finding new solutions to problems posed, and finding new surprising combinations of existing ideas), stochastic pattern mining (finding new surprising combinations...), and blending (exploratory fusion of existing concepts and relationships to form new ones).

Samsonovich:

There is a number of mechanisms of creation of new schemas in our architecture. A process of new schema creation starts with a prototype for a schema (called a "dream" or a "hypothesis"). The prototype may be functional in imagery even at the stage when the schema is not available. One possibility to complete a new schema is to combine existing schemas. When created, a new schema needs to be tested.

Wang:

Creativity occurs in NARS when the derived task/belief/concept does not exists in the system previously. There is no separate mechanism responsible for it, and whether an item is "creative" is sometimes a matter of degree.

- Does your AI system involve any analogue of the human notion of "f**eelings**." If so, what are the similarities and differences?

Franklin:

IDA doesn't learn and has handcrafted motivations in the form of drives attached to the behavior net. No feelings. LIDA, on the other hand, has feelings implemented in the perceptual module that are carried, as part of the common currency, throughout the whole system. In particular, feelings modulate learning and provide motivation.

Goertzel:

Novamente has "internal sensors" that serve as the roots of feelings – e.g. an internal sensor measuring the amount of knowledge recently gained; one measuring the amount of reward given to the system by its teachers; etc. "Feelings" in a more macro-level psychological sense are then broader patterns of activation, triggered by and/or closely associated with these internal sensors.

Samsonovich:

Yes, our architecture has an emotional cognitive map inspired by the human system of emotional values. The primary source for emergence of feelings in our architecture is the reward and punishment system (one of the 8 components). It includes a set of internal stimuli that are associated with selected schemas.

Wang:

In NARS, feeling and emotions comes out of the system's desirability on both external entities/events and internal status/situations. They will influence the internal control and the external communication of the system. As human feelings, they will be experience-grounded and context-sensitive, but since the system's experience will be different from human experience, we cannot expect the system to have the same feeling about a given event or status as a typical human being.

- How will your AI system give rise to a dynamic "**self-model**"? How will this model be represented and what learning methods will create and update it?

Franklin:

IDA has no self-model. In LIDA a self-model can be expected to emerge from autobiographical memory, a part of declarative memory.

Goertzel:

This has not been observed yet in practice, but a robust and complex self-model is expected to emerge in a Novamente system via the combined action of multiple cognitive mechanisms. The system must observe its own actions (e.g. in the AGISim simulation world), and the reactions of other systems to its actions, and construct a model of itself accordingly. Based on this self-model, it directs its ongoing actions, and hence obtains new information with which to improve its self-model. Inference and evolutionary learning both play key roles here, in terms of solving the cognitive problem of guessing "what compact model might describe what I am, in order to explain why I act the way I do." "Self" will not be a single concept represented by a single node, but rather a large set of nodes and links coordinated by a number of overlapping habitual patterns of activation.

Samsonovich:

The self is implemented in our architecture in multiple instances, as mental states. In a sense, each mental state can be called a self-model. The dynamics of mental states conforms to a set of fundamental beliefs (self axioms) that are hard-coded in the driving engine. E.g., the self must be consistent over time. Finally, the personal system of values developed by the cognitive map module is an essential aspect of the Self.

Wang:

The system will have a self-concept, with beliefs about what its status, needs, abilities, and so on. This concept will be handled as the other concepts.

Advances in Artificial General Intelligence: Concepts, Architectures and Algorithms
B. Goertzel and P. Wang (Eds.)
IOS Press, 2007
© 2007 The authors and IOS Press. All rights reserved.

A Foundational Architecture for Artificial General Intelligence

Stan FRANKLIN
Computer Science Department & Institute for Intelligent Systems, The University of Memphis

Abstract. Implementing and fleshing out a number of psychological and neuroscience theories of cognition, the LIDA conceptual model aims at being a cognitive "theory of everything." With modules or processes for perception, working memory, episodic memories, "consciousness," procedural memory, action selection, perceptual learning, episodic learning, deliberation, volition, and non-routine problem solving, the LIDA model is ideally suited to provide a working ontology that would allow for the discussion, design, and comparison of AGI systems. The LIDA architecture is based on the LIDA cognitive cycle, a sort of "cognitive atom." The more elementary cognitive modules and processes play a role in each cognitive cycle. Higher-level processes are performed over multiple cycles. In addition to giving a quick overview of the LIDA conceptual model, and its underlying computational technology, we argue for the LIDA architecture's role as a foundational architecture for an AGI. Finally, lessons For AGI researchers drawn from the model and its architecture are discussed.

Introduction

Early AI researchers aimed at what was later called "strong AI," the simulation of human level intelligence. One of AI's founders, Herbert Simon, claimed (circa 1957) that "… there are now in the world machines that think, that learn and that create." He went on to predict that with 10 years a computer would beat a grandmaster at chess, would prove an "important new mathematical theorem, and would write music of "considerable aesthetic value." Science fiction writer Arthur C. Clarke predicted that, "[AI] technology will become sufficiently advanced that it will be indistinguishable from magic" [1]. AI research had as its goal the simulation of human-like intelligence.

Within a decade of so, it became abundantly clear that the problems AI had to overcome for this "strong AI" to become a reality were immense, perhaps intractable. As a result, AI researchers concentrated on "weak AI" (now often referred to as "narrow AI"), the development of AI systems that dealt intelligently with a single narrow domain. An ultimate goal of artificial human-level intelligence was spoken of less and less.

As the decades passed, narrow AI enjoyed considerable success. A killer application, knowledge-based expert systems, came on board. Two of Simon's predictions were belatedly fulfilled. In May of 1997, Deep Blue defeated grandmaster and world chess champion Garry Kasparov. Later that year, the sixty-year-old Robbins conjecture in mathematics was proved by a general-purpose, automatic theorem-prover [2]. Narrow AI had come of age.

More recently, and perhaps as a result, signs of a renewed interest in a more human-like, general artificial intelligence began to appear. An IEEE Technical Committee on Autonomous Mental Developmental was formed, aimed at human-like learning for software agents and mobile robots. Motivated by the human autonomic nervous system, IBM introduced self-managed computer systems, called autonomic systems, designed to configure themselves, to heal themselves, to optimize their performance, and to protect themselves from attacks. In April of 2004, DARPA, the Defense Advanced Research Projects Agency sponsored a workshop on Self-Aware Computer Systems which led to a call for proposals to create such systems. AAAI-06 had a special technical track on integrated intelligent capabilities, inviting papers that highlight the integration of multiple components in achieving intelligent behavior. All these are trends toward developing an artificial, human-like, general intelligence.

The next major step in this direction was the May 2006 AGIRI Workshop, of which this volume is essentially a proceedings. The term AGI, artificial general intelligence, was introduced as a modern successor to the earlier strong AI.

Artificial General Intelligence

What is artificial general intelligence? The AGIRI website lists several features, describing machines

- with human-level, and even superhuman, intelligence.
- that generalize their knowledge across different domains.
- that reflect on themselves.
- and that create fundamental innovations and insights.

Even strong AI wouldn't push for this much, and this general, an intelligence. Can there be such an artificial general intelligence? I think there can be, but that it can't be done with a brain in a vat, with humans providing input and utilizing computational output. Well, if it can't be a brain in a vat, what does it have to be? Where can one hope to create artificial general intelligence (AGI)?

Autonomous Agents

Perhaps it can be created as an autonomous agent. And what's an autonomous agent? Biological examples include humans and most (all?) animals. Artificial autonomous agent can include software agents such as the bots that serve to populate Google's databases, and our IDA that does personnel work for the U.S. Navy ([3,4]). Other such examples include some mobile robots and computer viruses. But what do I mean by *autonomous agent*? Here's a definition [5]. It's a system that

- is embedded in an environment,
- is a part of that environment,
- which it senses,
- and acts on,
- over time,
- in pursuit of its own agenda (no human directs its choice of actions),
- so that its actions may affect its future sensing (it's structurally coupled to its environment ([6,7])).

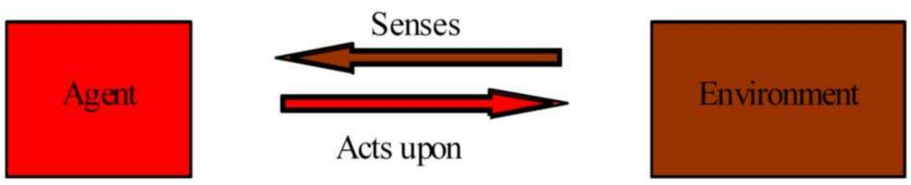


Figure 1. An Agent in its Environment.

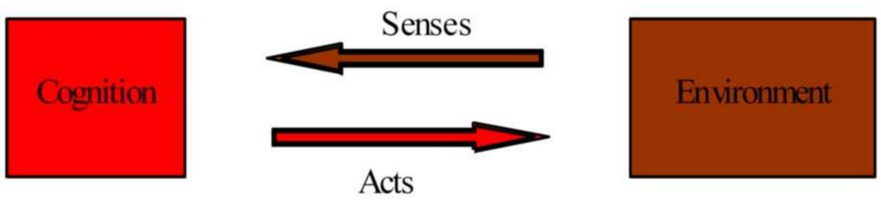


Figure 2. Cognition.

The earlier features of an autonomous agent listed in the definition are typically accepted in the community of agent researchers. The final feature is needed to distinguish an autonomous agent from other kinds of software, like a check-writing program that reads a personnel database once a week and produces a check for each employee.

Why must an AGI system be an autonomous agent? In order for an AGI system to generalize its knowledge across different, and likely novel, domains, it will have to learn. Learning requires sensing, and often acting. An autonomous agent is a suitable vehicle for learning [8], particularly for human-like learning ([9–11]).

An Agent in Its Environment

So the picture I'm going to propose as the beginning of a suggested ontology for AGI research is developed from that of an agent that senses its environment and acts on it, over time, in pursuit of its own agenda.

In order to do all this, it must have built in *sensors* with which to sense, it must have *effectors* with which to act, and it must have *primitive motivators* (which I call *drives*), which motivate its actions. Without motivation, the agent wouldn't do anything. Sensors, effectors and drives are primitives that must be built into, or evolved into, any agent.

Cognition

Next we replace the agent box in Fig. 1 with a box called Cognition (see Fig. 2).

For any autonomous agent, including we humans, the one significant, and constantly recurring question is "what to do next" ([12], Chapter 16). Cognition is the term we will use for this unending succession of deciding "what to do next," in some sense the only question there is. We humans face that question every moment of our existence, as must our artificial autonomous agents. In humans, such selected actions in-

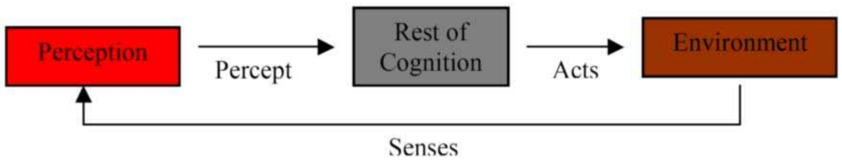


Figure 3. Perception.

clude movements in the hands, turning of the head, saccades of the eyes, movements of the lips and tongue when speaking, and all sorts of other actions.

Do note that this use of the term cognition is broader than the way the psychologists use it. When they talk about cognition, they normally don't include perception or the process of actually taking the action.

In what follows, we'll break out modules and processes, one or two at a time, from the Cognition box, replacing it with a gray Rest of Cognition box, and talk about such modules and processes individually. Each such module or process will become part of the developing ontology, hopefully acting as a foundation for AGI. I know you'll think this won't sound anything like AGI. It's far from general intelligence, but in my view this is where we have to start. This simple ontology will lead us to the beginnings of a foundational architecture for AGI.

Perception

Let's first break out *perception* (see Fig. 3), that is, the process of assigning of meaning to incoming sensory data. All this sensory data coming in must, somehow, be made meaningful to the agent itself. The agent must recognize individual objects, must classify them, and must note relations that exist between objects. The agent must make sense of the "scene" in front of it, perhaps including a visual scene, an auditory scene, an olfactory scene, etc. Perception is the process of agent making sense of its world.

And, what does meaning mean? How do you measure meaning? In my view, it's best measured by how well the meaning assists the agent in deciding what to do next, in action selection. This process of assignment of meaning can be bottom-up, that is, drawn immediately from the sensation. It can also be top-down with older meanings coming back around in additional cycles and contributing to later meanings. Perception can also be top-down within a single cycle, in that it can look back more than once at the incoming sensory data. In the diagram of Fig. 3, we break out perception at one end, and have it construct a *percept* that sends the meaningful aspects of the scene forward. Top-down influences of both types may contribute to the percept, contributing to the sense of the scene.

Procedural Memory

Next we're going to break out *procedural memory* (see Fig. 4), by which I mean a repertoire of tasks (actions, procedures). Actions from this repertoire can be executed singly, in parallel, in series, and even in more complex streams of actions. In addition to the actions themselves, procedural memory might also keep track of the context in which the action may prove useful, as well as the expected result of performing the action.

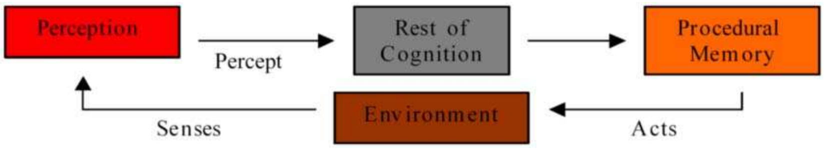


Figure 4. Procedural Memory.

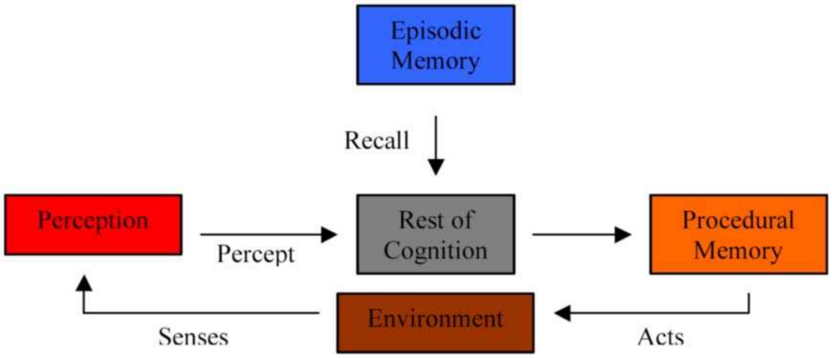


Figure 5. Episodic Memory.

But don't confuse procedural memory, which has to do with WHAT to do next, with sensory-motor memory, which keeps track of HOW to do it. How do I pick up a glass, turn it over and put it back? Deciding to pick it up is one thing; actually doing it is something different. These two kinds of memory should require different mechanisms.

Episodic Memory

Next we'll break out *episodic memory* (see Fig. 5), that is, the content-addressable, associative memory of events, of the what, the where and the when of the previous happening. These episodic memories may include semantic, locational, temporal, emotional, and causal aspects of the event.

Normally in humans, recall from episodic memory is accomplished by some kind of internal virtual reality. We build mental images, using them to partially re-live the event. These images may be visual, auditory, or whatever. Might such virtual imagery recall be possible, or useful, in artificial agents, I don't know. Animal cognition researchers often try to avoid controversy about animal consciousness by referring to "episodic-like" memory, defining it as the storing of the what, the when, and the where without any assumption of mental imaging ([13,14]).

Episodic memories come in several varieties. *Transient episodic memory* has a decay rate measured in hours or perhaps a day ([15,4,8,16,17]). Long-term episodic memories have the potential of storing information indefinitely. Long-term *declarative memory* includes *autobiographical memory*, the memory of events as described above, and *semantic memory*, the memory for facts.

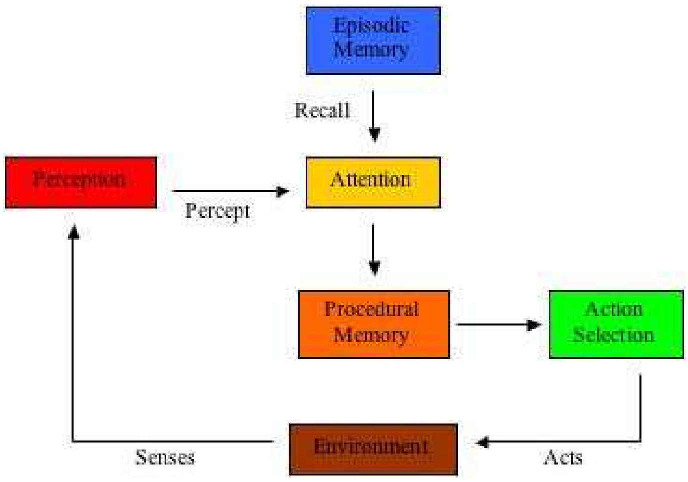


Figure 6. Attention and Action Selection.

Attention & Action Selection

In this section the gray "rest of cognition box" has disappeared, to be replaced by attention and action selection. *Attention* is the process that brings information built from perception and from episodic memory to consciousness. The global workspace theory of consciousness ([18–20]) postulates a competition for consciousness (see Cognition as Filtering below). The competition aims at selecting the most relevant, the most important, the most urgent, or the most insistent information to become conscious. This is a functional view of consciousness, and takes no stand on the possibility of subjective machine consciousness in an AGI [21].

The winning conscious information serves to recruit internal resources from which the next task is selected by the *action selection* mechanism. For an AGI such action selection must be quite sophisticated. In particular, it must be able to choose well between tasks serving different concurrent goals. It also must be able to bounce between seeking two such concurrent goals so as to take advantage of opportunities offered by the environment.

Cognition as Filtering

Following the *cognitive cycle* displayed in Fig. 6 above, we can usefully think of each step as a filtering process. An agent's sensory receptors filter all of the possible sensory data available in the environment, letting through only that to which the agent's sensors respond. Perception, as described above, is also a filtering process. Some sensory data coming in are ignored, while others are processed into possibly useful information, and become part of the percept that moves forward. The recall associations returned from a possibly huge episodic memory, accumulated over sometimes extremely long time periods, are also the result of a filtering process, What's wanted is information relevant to, and important for, the agent's current situation, including its goals. Hopefully, that's what comes out of this filtering process so far. Attention is yet another filtering process that decides what part of the recent percepts and episodic recall to bring to consciousness. Again the criteria for this filtering include relevance, importance, urgency, and insistence. Procedural memory then uses the contents of consciousness, what comes to attention, to recruit only those actions that might be possible and useful in the current situation, yet another filtering process. Our final filtering process is action selection, the process of choosing what single action to perform next.

The more complex the environment, and the more complex the agent, the more filtering is needed. One can think of the whole cognitive cycle, and even of cognition itself, as being essentially a complex, compound, filtering process.

Learning

Since, by its very nature, an AGI must learn, we next add several sorts of learning to the cognitive cycle (see Fig. 7 below). Our assumption is that the agent learns that to which it attends ([18] Section 5.5). Thus the learning arrows, in red, immerge from the Attention box. You'll see three different kinds of learning denoted here, though there are others. There's *perceptual learning*, the learning of new meanings, that is of objects, categories, relations, etc., or the reinforcement of existing meanings. The *episodic learning* of events, the what, the where and the when, is denoted by its encoding. Finally, *procedural learning* improves skills and/or to learns new skills.

A Foundational Architecture for AGI

So, if we're going to aim for an AGI, where do you look for it? How should we go about trying to build an AGI agent? In my view, if you want smart software, copy it after a human. That is, model the early AGI agents on what we know about human cognition. In the previous sections we've discussed modules and processes, derived from human cognition, that we believe must be included in any AGI architecture. Where do we go from there? One possibility is to dive right in and attempt to build a

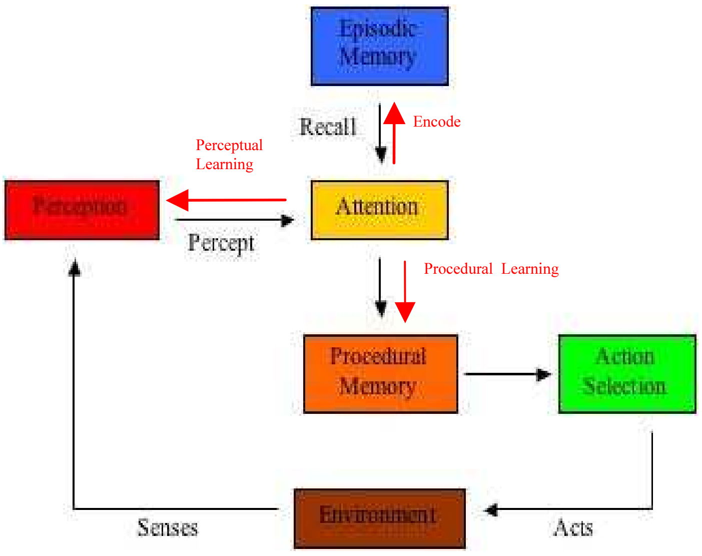


Figure 7. Learning.

full-blown AGI directly. This strategy, while surely ambitious, may well succeed. A second possible strategy might be to construct a sequence of increasingly complex, intelligent, and general, artificial agents, culminating in a true AGI. This second strategy may prove to be even more likely to succeed.

Here we suggest a means of enabling this second strategy by way of a common foundational architecture for each agent in the sequence. Such a foundational architecture would allow each successive agent to be built by adding higher-level cognitive processes to its predecessor. Let's assume, as we must ([22–25,9,10]), that learning via a developmental period must be an integral part of the life cycle of any AGI. The strategy suggested might allow what's learned by one robot to be initially incorporated into its immediate successor.

Any autonomous agent, and hence any AGI, must operate by means of a continuing iteration of cognitive cycles, the sense-cognize-act cycles described above. Each such cognitive cycle acts as a cognitive moment, an atom of cognition, in that each higher-level cognitive process is performed via the execution of a sequence of cognitive cycles. Higher-level cognitive processes are built of these cognitive cycles as cognitive atoms. Thus, a foundational architecture for AGI must implement a cognitive cycle to be continually iterated, and must provide mechanisms for building higher-level cognitive processes composed of sequences of these cognitive cycles. The LIDA architecture, to be described next, accomplishes both.

The LIDA Architecture

IDA denotes a conceptual and computational model of human cognition. LIDA, short for Learning IDA, denotes another such model with learning added. Let's start with a brief description of IDA.

The US Navy has about 350,000 sailors. As each sailor comes to the end of a certain tour of duty, he or she needs a new billet, a new job. The Navy employs some 300 detailers, as they call them, personnel officers who assign these new billets. A detailer dialogs with sailors, usually over the telephone, but sometime by email. These detailers read personnel data from a sailor's record in a Navy personnel database for items bearing on qualifications. They check job requisition lists in another Navy database to see what jobs will come available and when. They enforce the Navy's policies and try to adhere to the sailors' wishes, as well as looking to the needs of the particular job. Eventually, the detailer offers one, two or, rarely, three jobs to the sailor. Some back and forth negotiations ensue, involving several communications. Hopefully the sailor agrees to takes a job offered by the detailer. If not, the detailer simply assigns one.

IDA, an acronym for Intelligent Distribution[1] Agent, is an autonomous software agent, which automates the tasks of the detailers as described in the previous paragraph ([25,26]). Built with Navy funding, IDA does just what a human detailer does. In particular, she communicates with sailors in natural language, in English, though by email rather than telephone. The sailor writes anyway he or she wants to write. There's no prescribed protocol or format, no form to fill out. IDA understands what the sailor writes in the sense of knowing how to pick out relevant and important pieces of information from the email message, and what to do with it. IDA is implemented, up and running, and tested to the Navy's satisfaction.

[1] Distribution is the Navy's name for the process of assigning new jobs to sailors at the end of a tour of duty.

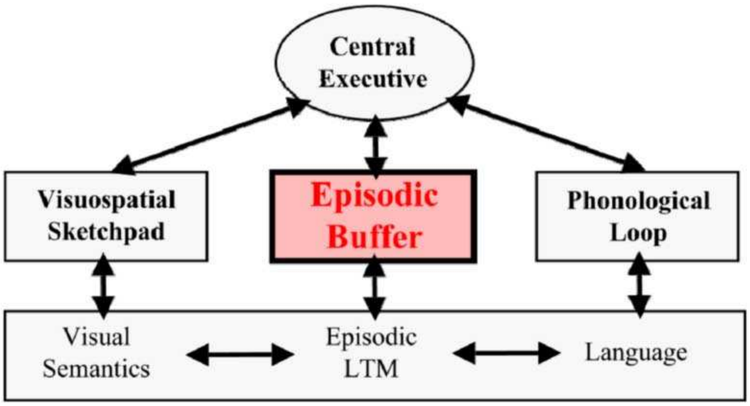


Figure 8. Working Memory.

To accomplish the tasks of a human detailer, IDA employs a number of higher-level cognitive processes. These include constraint satisfaction [27], deliberation ([28,29]), sophisticated action selection [30] and volition [29].

Both in its cognitive cycle and its implementation of higher-level cognitive processes, IDA, and its learning extension LIDA, implement a number of mostly psychological theories of cognition. We'll very briefly describe each, and its role in the LIDA architecture.

Over the past couple of decades, research in AI, and more generally cognitive science, has moved towards situated or embodied cognition [31]. The idea is that cognition should be studied in the context of an autonomous agent situated within an environment. Being an autonomous software agent, IDA is embodied [32]. Similarly, software agents, autonomous robots or AGI's built on the foundation of a LIDA architecture would be embodied.

Barsalou, in his theory of perceptual symbol systems [33] postulates that there are no amodal symbols involved in human cognition. Rather, *all* such information is represented by perceptual symbols. Put another way, all cognitive symbols are ultimately grounded in perception [34]. The LIDA architecture represents perceptual entities, objects, categories, relations, etc., using nodes and links in a slipnet [35]. These serve as perceptual symbols acting as the common currency for information throughout the various modules of the LIDA architecture.

In cognitive psychology the term working memory refers to a theoretical framework specifying and describing structures and processes used for temporarily storing and manipulating information [36]. Among these structures are the visuospacial sketchpad, the phonological loop, and a central executive responsible for the integration of information. More recent working memory structures include a consciousness mechanism [37] and the episodic buffer [38] (see Fig. 8). All of the various modules and processes working memory are implemented in the LIDA architecture, mostly in its perceptual module and its workspace (see below) [39].

Glenberg's theory [40] stresses the importance of patterns of behavior to conceptualization and to understanding. For example, an object is understood via its affordances [41]. In the LIDA architecture templates for such pattern of behavior are found in perceptual memory. Their instantiations as sequences of actions contribute to perceptual learning, including conceptualization, leading to further understanding.

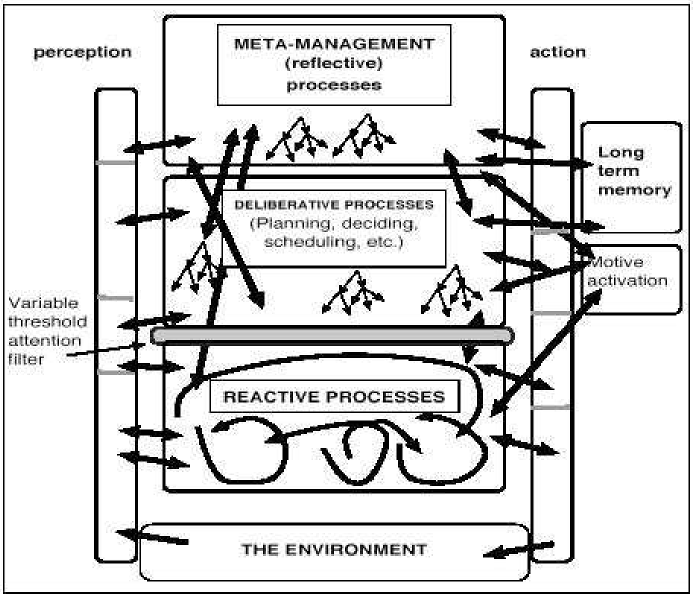

perception
META-MANAGEMENT (reflective) processes
action
Long term memory
DELIBERATIVE PROCESSES (Planning, deciding, scheduling, etc.)
Motive activation
Variable threshold attention filter
REACTIVE PROCESSES
THE ENVIRONMENT

Figure 9. Sloman's Architecture.

The long-term working memory of Ericsson and Kinstch [42] is incorporated into LIDA's workspace (see below), in which local associations recalled from episodic memory are combined with percepts to produce higher-level perceptual structures. In an AGI this workspace would include the various working memory structures mentioned above.

By far the single most significant influence on the LIDA architecture from cognitive psychology came from Baars' global workspace theory (GWT) of consciousness and cognition ([18,19,29,39]). GWT postulates attention, bringing important, relevant information to consciousness (the global workspace). Its contents are then broadcast to the entire system in order to recruit internal resources with which to deal appropriately with the current situation. The LIDA architecture implements precisely this function, as will become clear during the discussion of the LIDA cognitive cycle below.

Finally, the LIDA architecture can be thought of a fleshing out and an implementation of Sloman's architecture for a human-like agent [28]. One can construct a concordance between most of the various modules and processes shown in Fig. 9 and corresponding modules and processes of the LIDA cognitive as shown in Fig. 10 below. Those that won't fit in to such a concordance correspond to higher-level, multi-cyclic cognitive processes in LIDA (see Multi-cyclic Cognitive Processes below). Sloman's meta-management process, that is, what the psychologist call metacognition, has not yet been designed for the LIDA model, but it certainly can be.

The LIDA Cognitive Cycle

LIDA operates as any autonomous must, with a continuously iterating cognitive cycle. Higher-level cognitive processes are composed of sequences of several or many of these cognitive cycles. Such higher-level cognitive processes might include deliberation, volition, problem solving, and metacognition.

Let's take a quick, guided tour through LIDA's cognitive cycle, which is based on Fig. 7 above. Figure 10 below will provide a useful map for our tour. Note, that this cognitive cycle is highly complex, and yet all of this must be accomplished in every cognitive moment. Computational resources may well prove an issue.

Beginning at the upper left of Fig. 10, we see stimuli coming in from both the internal and the external environment. Recall that, by definition, every autonomous agent is a *part* of its environment. LIDA is modeled after humans; we have to deal with both external and internal stimuli. Any AGI will likely have to also do so.

In *Sensory Memory* (SM) one would find the sensors themselves and primitive, that is, built-in, feature detectors. It would also include early learned, and therefore not primitive, feature detectors that provide the beginnings of understanding of the stimuli. Note that information from SM goes both to Perceptual Associative Memory, which we'll discuss next, and to the Effectors via the SMA (sensory-motor automatisms). In the later role, SM is crucially involved in quickly providing the kind of precise spatial, temporal, egocentric information that permit such actions as successfully hitting an oncoming fast ball, or even the grasping of a cup. Such SMA's in humans operate on their own direct sensory-motor cycles at about five times the rate of the larger LIDA cognitive cycle.

From SM, information travels to *Perceptual Associative Memory* (PAM), which we implement as a slipnet [35]. Here the next stage of constructing meanings occur, that is the recognition of further features, of objects, and of categories. Passing activation brings some nodes and links over threshold, and thus into the *percept*.

The LIDA cognitive cycle includes two episodic memory modules, the short-term *Transient Episodic Memory* (TEM), and the potentially long-term *Declarative Memory* (DM) ([15,4,8,16]). Recording such information as where I parked my car in the garage this morning, TEM encodings decay in humans within hours or a day. DM encodings

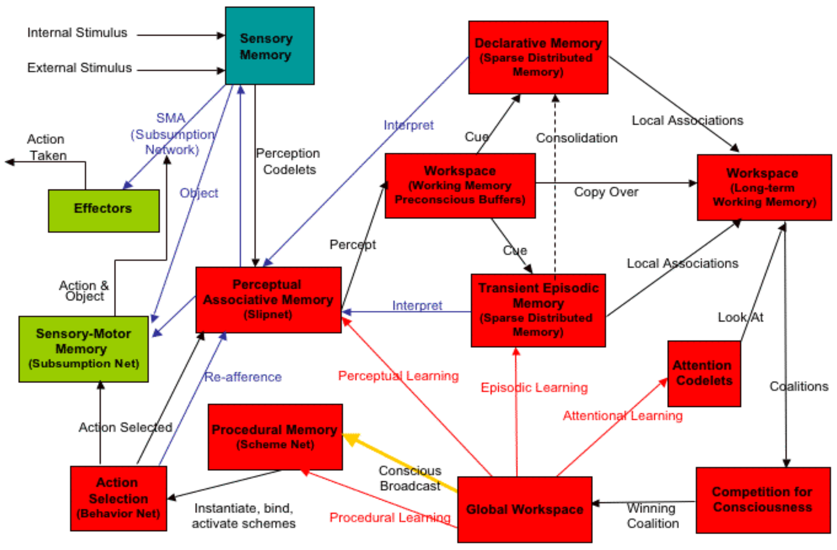

Internal Stimulus
External Stimulus
Sensory Memory
Declarative Memory (Sparse Distributed Memory)
SMA (Subsumption Network)
Action Taken
Effectors
Object
Perception Codelets
Interpret
Cue
Consolidation
Local Associations
Workspace (Working Memory Preconscious Buffers)
Copy Over
Workspace (Long-term Working Memory)
Percept
Cue
Local Associations
Action & Object
Perceptual Associative Memory (Slipnet)
Interpret
Transient Episodic Memory (Sparse Distributed Memory)
Look At
Sensory-Motor Memory (Subsumption Net)
Re-afference
Perceptual Learning
Episodic Learning
Attention Codelets
Coalitions
Attentional Learning
Action Selected
Procedural Memory (Scheme Net)
Conscious Broadcast
Action Selection (Behavior Net)
Instantiate, bind, activate schemes
Procedural Learning
Global Workspace
Winning Coalition
Competition for Consciousness

Figure 10. The LIDA Cognitive Cycle.

only occur through offline consolidation from TEM. Though they can decay away, when sufficiently reinforced DM encodings can last a lifetime. Both episodic memories are computationally implemented using a modified sparse distributed memory ([43–45]).

The percept produced by PAM (described two paragraphs above) is moved into the *Workspace*, an amalgam of the preconscious working memory buffers and long-term working memory (in Fig. 10 the Workspace is split into two boxes). Here additional, more relative, less precise, scene understanding structures are built. As well as the current percept, the Workspace also contains previous percepts and recent local associations recalled from both TEM and DM, all in various stages of decaying away. These contents of the Workspace serve to cue TEM and DM for current local associations. An understanding of the current scene is produced in the Workspace using additional, quick, two-way communication, that is, including down-stream communication, with TEM, DM, PAM and even SM.

Next, the *Attention Codelets*,[2] whose job it is to bring relevant and important information to consciousness, come into play. An attention codelet has its own special interests, to which it wishes to draw attention. Each attention codelet searches the workspace for items (objects, relations, situations) of interest, and creates coalitions[3] of these items if it finds them.

These coalitions move into the Global Workspace where there's a competition for consciousness. This competition constitutes the final filtering of input. The idea is to attend to filter most relevant, the most important, the most urgent, the most insistent aspects of the current situation. Once the competition for consciousness is resolved, GWT call for a global broadcast of the contents of consciousness.

Aside from learning, which we'll discuss later, the major recipient of the global broadcast is *Procedural Memory* (PM), which we implement as a scheme net modeled after the schema mechanism [46]. M uses the contents of the global broadcast to pick out those possible actions that might be relevant to the current situation. Each scheme in PM is a template for an action together with its context and result. The schemes that might be relevant, that is, those whose context and/or results intersect with the contents of the global broadcast, including goals, instantiate themselves and bind their variables with information from the broadcast.

These instantiated schemes then go to Action Selection (AS), which is implemented as a behavior net ([47,30]), a very sophisticated kind of action selection mechanism. In AS, instantiated schemes compete to be the single action selected, possibly a compound of sub-actions in parallel. Over multiple cognitive cycles, AS may select a sequence of actions to accomplish a given goal. It might also bounce opportunistically between sequences of actions serving different goals.

The single action chosen during a given cognitive cycle is sent, along with the object(s) upon which it is to act, to Sensory-Motor Memory (S-MM), which contains procedures for actually performing the selected action, the so called sensory-motor automatisms. Our representation for these sensory-motor automatisms is as yet undecided, but we're leaning toward a net built from subsumption networks [48].

In our tour through the LIDA cognitive cycle we postponed a discussion learning, which we'll take up now. Our basic premise is that we learn that to which we attend

[2] Taken from the Copycat Architecture (Hofstadter and Mitchell 1995), "codelet" refers to a small, special-purpose piece of computer code, often running as a separate thread. They implement the processors of GWT (Baars 1988). There are many different varieties of codelet in the LIDA model.

[3] The term "coalition" comes from GWT, where it always refers to a coalition of processors.

([18] pp. 213–214). Thus learning occurs as a consequence of, or at least in conjunction with, the conscious broadcast from the Global Workspace. Learning is modulated by affect following an inverted U curve. Learning is strengthened as affect increases up to a point. After that the affect begins to interfere and the learning rate diminishes with further increases in affect ([49,50]).

The LIDA cognitive cycle includes four types of learning, three of which were discussed earlier in the chapter [9]. The perceptual learning of object, categories, relations, etc., takes place in PAM [8]. Episodic learning of what, where and when are encoded in TEM, while procedural learning of tasks takes place in PM [10]. The hitherto unmentioned form of learning is attentional learning, the learning of what to attend, which takes place in the Attention Codelets. We know little about attentional learning, which is an object of current research.

Each of these types of learning has its selectionist and its instructionalist form [51]. Selectionist learning reinforces existing memory traces positively or negatively. Instructionalist learning adds new entities to the various memories, often by altering or combining existing entities.

Following our strategy, mentioned above, of producing smart software by copying humans, the LIDA cognitive cycle was modeled after what we hypothesize happens in humans ([39,4,8,16]). Though asynchronous, each cognitive cycle runs in about 200 milliseconds. But they can cascade, so a new cycle can begin while earlier cycles are completing. As a consequence of this cascading, the rate of this cognitive cycle processing is five to ten cycles per second. Though asynchronous, the seriality of consciousness must be preserved. Though none of it is conclusive, there considerable evidence from neuroscience suggestive or, or supportive of, these cognitive cycles in nervous systems ([52–55]).

Multi-Cyclic Cognitive Processes

In the LIDA model cognitive cycles are the atoms out of which higher-level cognitive processes are built. Here we'll briefly describe several of these higher-level processes: deliberation, volition, atomization, non-routine problem solving, metacognition and self-awareness. Each of these is a multi-cyclic process that can be implemented over multiple cognitive cycles using the LIDA architecture as a foundation. Let's take them up one at a time, beginning with deliberation.

Deliberation refers to such activities as planning, deciding, scheduling, etc. that require one to consciously think about an issue. Suppose I want to drive from a new location in a city I know to the airport. It will be a route I've never taken, so I may imagine landmarks along the way, which turns to take and so, deliberate about how best to get there. When IDA thinks about whether she can get a sailor from a current job to a specific new job with leave time, training time, travel time and so forth all fitted in between, that's deliberation. This higher-level deliberative process takes place in IDA (and LIDA) over multiple cognitive cycles using behavior streams instantiated from PM into the behavior net (AS) [29].

As specified by GWT, conscious, volitional, decision-making, a kind of deliberation, is implemented via William James' ideomotor theory ([56,18,29]). Once again, *volition* uses an instantiated behavior stream over several cognitive cycles. For example, suppose that, being thirsty one morning, I consciously considered the possibilities of coffee, tea, and orange juice, weighing the advantages and disadvantages of each, perhaps by arguing with myself. My eventually deciding to drink tea is a volitional

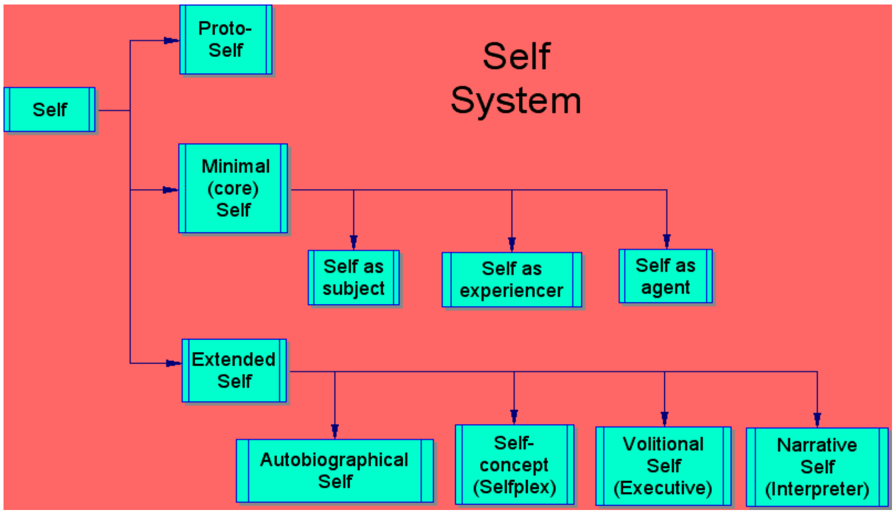


Figure 11. Various Selves.

decision, as opposed to my typing of this phrase, which was not consciously decided on ahead of time. IDA decides volitionally on which jobs to offer sailors.

How do we get from consciously going through all the steps of learning to drive an automobile, to the effortless, frequently unconscious, automatic actions of an experienced driver? We call this higher-level cognitive process *automization*, and implement it in the LIDA model via pandemonium theory ([57,58]). Once again automization is accomplished over multiple cognitive cycles using the LIDA architecture as a framework.

In the LIDA architecture Procedural Memory (PM) consists of templates for actions, including their contexts and expected results. Actions are selected from among action templates instantiated in response to a conscious broadcast. What if PM doesn't contain any action templates to be recruited to deal with the current situation? In this case *non-routine problem solving* would be required. The LIDA architecture serves as a foundation for an, as yet unpublished, non-routine problem solving algorithm based on an extension of partial order planning [59].

Defined by psychologists as thinking about thinking, *metacognition*[4] has, in recent years become of interest to AI researchers ([60,28,61]). There's even a website for Metacognition in Computation (www.cs.umd.edu/~anderson/MIC). Metacognition is often used to update a strategy. Suppose I think that I was too hard on my daughter in our interaction last night, and decide that next time I want to be more empathetic with her. That's an example of metacognition. After early, misguided attempts (see for example, [62]), we now know how to build metacognition as a collection of higher-level cognitive processes on a foundation of the LIDA architecture and its cognitive cycle. This work is currently in an early stage and not yet published.

Philosophers, psychologists and neuroscientists have defined and studied a number of varieties of selves ([63–67]) (see Fig. 11) Finally, it's possible to implement several of the various varieties of *self* as higher-level cognitive processes on a foundation of

[4] Sloman calls it meta-management (see Fig. 9).

the LIDA architecture. Again, this work has currently just begun and is as yet unpublished.

All of these and many, many more multi-cyclic processes can be built using the LIDA architecture's cognitive cycles as cognitive atoms. It's this possibility that supports the strategy of producing an AGI as a sequence of ever more intelligent, adaptable and versatile autonomous agents each containing the previous, and each based on the LIDA architecture.

Lessons for Building an AGI

Suppose we accept the strategy of building an AGI as the culmination of an increasing sequence of ever more intelligent and adaptable AGI agents, each built on the foundation of the LIDA architecture with its cognitive cycles as atoms. What general lessons can we learn as a result? Here are a few.

We must choose a suitable domain for our agent. A domain? A domain for an AGI agent? I thought an AGI was supposed to generalize. It certainly must generalize, but it's still an autonomous agent. Every such agent must come with built-in sensors, motivators, and effectors. That means the agent must have an environment on which to sense and act, that is, a domain. What is needed is a well-chosen domain from which it can generalize. This would entail a broad enough domain with a number of subdomains from which it can generalize. The successor of each agent in the sequence may be situated in a more inclusive domain, and may be provided with additional sensors, motivators and effectors.

In my view, an AGI agent is much too much to handcraft. By definition, it's supposed to generalize, that is, to add to its store of knowledge and skill. Therefore it must learn. And, how shall it learn? At least at the start, I suggest that it learn like a human, that we build-in human-like learning capabilities. Later on we, or it, may find better ways of learning. Let's note some principles of human learning that can be adapted to human-like learning in an AGI agent, and in its predecessors.

There's no learning from scratch, from a blank slate. For example, human infants come equipped to recognize faces. The practice of the more sophisticated machine learning research community is to build in whatever you can build in. This same principle should be followed when attempting to build an AGI. Learning, yes. Learning from scratch, no.

With trivial exceptions, we learn that to which we attend, and only that. The implication is that an AGI must come equipped with an attention mechanism, with some means of attending to relevant information. This implies the need for some form of functional consciousness, but not necessarily subjective consciousness [21].

Human learning is incremental and continual. It occurs at every moment, that is, during every cognitive cycle. And, it's unsupervised. Supervised machine learning typically involves a training period during which the agent is taught, and after which it no longer learns. In contrast, an AGI agent will need to learn incrementally and continually as human's do. Though such an agent may go through a developmental period of particularly intense learning, it must also be a "lifelong" learner.

Humans learn by trial and error, that is, by what we in AI call a generate-and-test process. The LIDA model hypothesizes that we learn potential new objects in PAM quickly and on the flimsiest of excuses [8]. This is the process of generation. New objects that are reinforced by being attended to survive, while others decay away. This is the testing process. All this is done incrementally and continually, that is, in every cognitive cycle. And, this perceptual learning by generate-and-test is not restricted to new objects, but applies to categories, relations, etc. Similar processes are in place for episodic and procedural learning as well. I suggest that such generate-and-test learning will be needed in AGI agents as well.

According to the LIDA model, much if not all of human memory is content addressable. We don't access an item in memory by knowing its address or index. Rather we access it using a portion of its content as a cue. Sensory memory is cued by the incoming content of the sensors. In PAM detected features allow us to access objects, categories, relations, etc. The contents of LIDA's workspace cue both episodic memories, TEM and DM, recalling prior events associated with the cue. Action templates in PM are cued by the contents of the conscious broadcast. Such content addressable memories must surely be a part of any AGI agent.

Above we distinguished and spoke of selectionist and instructionalist learning within the LIDA architecture. I suggest that an AGI agent must also learn in each of these learning methods in each of its learning modes, perceptual, episodic, and procedural. In summary, the LIDA model suggests that an AGI agent must initially be copied after humans, must have a rich and broad domain, must employ many multi-cyclic processes, and must be capable of using both learning methods in the several different modes of learning.

Questions for AGI Researchers

Must an AGI agent be functionally conscious? As noted above, the LIDA model suggests an affirmative answer. Though functional consciousness as derived from GWT may not prove necessary, I suspect some form of attentional mechanism will.

Must an AGI agent be phenomenally conscious? That is, must it have subjective experiences as we do? I think not. I suspect that we may be able to build an AGI agent that's not phenomenally conscious. However, subjective consciousness may prove necessary to deal with the problem of distinguishing perceived motion due to changes in the environment from perceived motion due to movement of the agent's sensors ([68,16]). Subjective consciousness provides an agent with a coherent, stable internal platform from which to perceive and act on its world. We may be pushed into trying to build AGI agents that are phenomenally conscious.

Must an AGI agent be capable of imagination? That is, must it be able to produce an internal virtual reality? Humans often deliberate in this way. An AGI agent must certainly be capable of deliberation. However, deliberation has been shown to be implementable without subjective consciousness ([8,21]).

Must an AGI agent come equipped with feelings? In humans, feelings include, for example, thirst and pain, as well as emotions[5] such as fear or shame [69]. In humans and animals, feelings implement motivation. It's feelings that drive us to do what we do ([70,71]). We think we select actions rationally, but such decisions, though surely influenced by facts and expected consequences, are ultimately in the service of feelings. And, feelings modulate our learning by increasing affect, as discussed above [69]. Must an AGI agent have artificial feelings to serve these purposes? I think not. There are other ways to modulate learning; there are other ways to implement drives as primitive motivators. Feelings have proved useful evolutionarily in complex dynamic envi-

[5] Following [69] we think of emotions as feelings with cognitive content. For example, one might be afraid of the rapidly approaching truck, the cognitive content.

ronments because they provide a lot of flexibility in learning and action selection. They may provide a good solution to these problems in AGI agents, or we may find a better way.

Acknowledgements

The author is indebted to the many members, past and present, of the Cognitive Computing Research Group at the University of Memphis. Their individual and join efforts contributed significantly to the ideas presented in this chapter. In particular, thanks are due to Sidney D'Mello for a graphic on which Fig. 10 is based, and to Michael Ferkin who contributed to the development of the suggested ontology.

References

[1] Clarke, A. C. 1962. Profiles of the Future; an Inquiry into the Limits of the Possible. New York: Harper & Row.
[2] McCune, W. 1997. Soltion of the Robbins Problem. *Journal of Automated Reasoning* 19:263–275.
[3] Franklin, S. 2001. Automating Human Information Agents. In *Practical Applications of Intelligent Agents*, ed. Z. Chen, and L. C. Jain. Berlin: Springer-Verlag.
[4] Franklin, S. 2005a. A "Consciousness" Based Architecture for a Functioning Mind. In *Visions of Mind*, ed. D. N. Davis. Hershey, PA: Information Science Publishing.
[5] Franklin, S., and A. C. Graesser. 1997. Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. In *Intelligent Agents III*. Berlin: Springer Verlag.
[6] Maturana, H. R. 1975. The Organization of the Living: A Theory of the Living Organization. *International Journal of Man-Machine Studies* 7:313–332.
[7] Maturana, R. H. and F. J. Varela. 1980. *Autopoiesis and Cognition: The Realization of the Living, Dordrecht*. Netherlands: Reidel.
[8] Franklin, S. 2005b. Perceptual Memory and Learning: Recognizing, Categorizing, and Relating. Symposium on Developmental Robotics. American Association for Artificial Intelligence (AAAI). Stanford University, Palo Alto CA, USA. March 21–23, 2005.
[9] D'Mello, S. K., S. Franklin, U. Ramamurthy, and B. J. Baars; 2006. A Cognitive Science Based Machine Learning Architecture. AAAI 2006 Spring Symposium Series. American Association for Artificial Intelligence. Stanford University, Palo Alto, California, USA. March.
[10] D'Mello, S. K., U. Ramamurthy, A. Negatu, and S. Franklin. 2006. A Procedural Learning Mechanism for Novel Skill Acquisition. In *Workshop on Motor Development*: *Proceeding of Adaptation in Artificial and Biological Systems, AISB'06*, vol. 1, ed. T. Kovacs, and J. A. R. Marshall. Bristol, England: Society for the Study of Artificial Intelligence and the Simulation of Behaviour; April 2006.
[11] Ramamurthy, U., S. K. D'Mello, and S. Franklin. 2006. LIDA: A Computational Model of Global Workspace Theory and Developmental Learning. BICS 2006: Brain Inspired Cognitive Systems. October 2006.
[12] Franklin, S. 1995. *Artificial Minds*. Cambridge MA: MIT Press.
[13] Clayton, N. S., D. P. Griffiths, and A. Dickinson. 2000. Declarative and episodic-like memory in animals: Personal musings of a scrub jay or When did I hide that worm over there? In *The Evolution of Cognition*, ed. C. M. Heyes, and L. Huber. Cambridge, MA: MIT Press.
[14] Ferkin, M. H., A. Combs, J. delBarco-Trillo, A. A. Pierce, and S. Franklin. in review. Episodic-like memory in meadow voles, *Microtus pennsylvanicus*. *Animal Behavior*.
[15] Conway, M. A. 2001. Sensory-perceptual episodic memory and its context: autobiographical memory. *Philos. Trans. R. Soc. Lond B.* 356:1375–1384.
[16] Franklin, S. 2005c. Evolutionary Pressures and a Stable World for Animals and Robots: A Commentary on Merker. *Consciousness and Cognition* 14:115–118.
[17] Franklin, S., B. J. Baars, U. Ramamurthy, and M. Ventura. 2005. The Role of Consciousness in Memory. *Brains, Minds and Media* 1:1–38, pdf.
[18] Baars, B. J. 1988. *A Cognitive Theory of Consciousness*. Cambridge: Cambridge University Press.
[19] Baars, B. J. 1997. *In the Theater of Consciousness*. Oxford: Oxford University Press.

[20] Baars, B. J. 2002. The conscious access hypothesis: origins and recent evidence. *Trends in Cognitive Science* 6:47–52.
[21] Franklin, S. 2003. IDA: A Conscious Artifact? *Journal of Consciousness Studies* 10:47–66.
[22] Posner, M. I. 1982. Cumulative Development of Attentional Theory. *American Psychologist* 37: 168–179.
[23] Franklin, S. 2000a. Learning in "Conscious" Software Agents. In *Workshop on Development and Learning*, ed. J. Wang. Michigan State University; East Lansing, Michigan, USA: NSF; DARPA; April 5–7, 2000.
[24] Weng, J. 2004. Developmental Robotics: Theory and Experiments. *International Journal of Humanoid Robotics* 1:119–234.
[25] Franklin, S., A. Kelemen, and L. McCauley. 1998. IDA: A Cognitive Agent Architecture. In *IEEE Conf on Systems, Man and Cybernetics*. IEEE Press.
[26] McCauley, L., and S. Franklin. 2002. A Large-Scale Multi-Agent System for Navy Personnel Distribution. *Connection Science* 14:371–385.
[27] Kelemen, A., S. Franklin, and Y. Liang. 2005. Constraint Satisfaction in "Conscious" Software Agents – A Practical Application. *Applied Artificial Intelligence* 19:491–514.
[28] Sloman, A. 1999. What Sort of Architecture is Required for a Human-like Agent? In *Foundations of Rational Agency*, ed. M. Wooldridge, and A. S. Rao. Dordrecht, Netherlands: Kluwer Academic Publishers.
[29] Franklin, S. 2000b. Deliberation and Voluntary Action in 'Conscious' Software Agents. *Neural Network World* 10:505–521.
[30] Negatu, A., and S. Franklin. 2002. An action selection mechanism for 'conscious' software agents. *Cognitive Science Quarterly* 2:363–386.
[31] Varela, F. J., E. Thompson, and E. Rosch. 1991. *The Embodied Mind*. Cambridge, MA: MIT Press.
[32] Franklin, S. 1997. Autonomous Agents as Embodied AI. *Cybernetics and Systems.* 28:499–520.
[33] Barsalou, L. W. 1999. Perceptual symbol systems. *Behavioral and Brain Sciences* 22:577–609.
[34] Harnad, S. 1990. The Symbol Grounding Problem. *Physica D* 42:335–346.
[35] Hofstadter, D. R., and M. Mitchell. 1995. The Copycat Project: A model of mental fluidity and analogy-making. In *Advances in connectionist and neural computation theory, Vol. 2: logical connections*, ed. K. J. Holyoak, and J. A. Barnden. Norwood N.J.: Ablex.
[36] Baddeley, A. D., and G. J. Hitch. 1974. Working memory. In *The Psychology of Learning and Motivation*, ed. G. A. Bower. New York: Academic Press.
[37] Baddeley, A. 1992. Consciousness and Working Memory. *Consciousness and Cognition* 1:3–6.
[38] Baddeley, A. D. 2000. The episodic buffer: a new component of working memory? *Trends in Cognitive Science* 4:417–423.
[39] Baars, B. J., and S. Franklin. 2003. How conscious experience and working memory interact. *Trends in Cognitive Science* 7:166–172.
[40] Glenberg, A. M. 1997. What memory is for. *Behavioral and Brain Sciences* 20:1–19.
[41] Gibson, J. J. 1979. *The Ecological Approach to Visual Perception*. Mahwah, New Jersey: Lawrence Erlbaum Associates.
[42] Ericsson, K. A., and W. Kintsch. Ericsson, K. A., and W. Kintsch. 1995. Long-term working memory. Psychological Review 102:211–245. Long-term working memory. Psychological Review 102:211–245.
[43] Kanerva, P. 1988. Sparse Distributed Memory. Cambridge MA: The MIT Press.
[44] Ramamurthy, U., S. K. D'Mello, and S. Franklin. 2004. Modified Sparse Distributed Memory as Transient Episodic Memory for Cognitive Software Agents. In *Proceedings of the International Conference on Sstems, Man and Cybernetics*. Piscataway, NJ: IEEE.
[45] D'Mello, S. K., U. Ramamurthy, and S. Franklin. 2005. Encoding and Retrieval Efficiency of Episodic Data in a Modified Sparse Distributed Memory System. In *Proceedings of the 27th Annual Meeting of the Cognitive Science Society. Stresa, Italy.*
[46] Drescher, G. L. 1991. *Made-Up Minds: A Constructivist Approach to Artificial Intelligence*. Cambridge, MA: MIT Press.
[47] Maes, P. 1989. How to do the right thing. *Connection Science* 1:291–323.
[48] Brooks, R. A. 1991. How to build complete creatures rather than isolated cognitive simulators. In *Architectures for Intelligence*, ed. K. VanLehn. Hillsdale, NJ: Lawrence Erlbaum Associates.
[49] Belavkin, R. V. 2001. Modelling the inverted-U effect with ACT-R. In *Proceedings of the 2001 Fourth International Conference on Cognitive Modeling*, ed. E. M. Altmann, W. D. Gray, A. Cleeremans, and C. D. Schunn. Hillsdale, NJ: Lawrence Erlbaum Associates.
[50] Cochran, R. E., F. J. Lee, and E. Chown. 2006. Modeling Emotion: Arousal's Impact on memory. In *Proceedings of the 28th Annual Conference of the Cognitive Science Society (pp. 1133-1138). Vancouver, British Columbia, Canada*. Vancouver, British Columbia, Canada.
[51] Edelman, G. M. 1987. *Neural Darwinism*. New York: Basic Books.

[52] Lehmann, D., H. Ozaki, and I. Pal. 1987. EEG alpha map series: brain micro-states by space-oriented adaptive segmentation. *Electroencephalogr. Clin. Neurophysiol.* 67:271–288.

[53] Lehmann, D., W. K. Strik, B. Henggeler, T. Koenig, and M. Koukkou. 1998. Brain electric microstates and momentary conscious mind states as building blocks of spontaneous thinking: I. Visual imagery and abstract thoughts. *Int. J. Psychophysiol.* 29:1–11.

[54] Halgren, E., C. Boujon, J. Clarke, C. Wang, and P. Chauvel. 2002. Rapid distributed fronto-parieto-occipital processing stages during working memory in humans. *Cerebral Cortex* 12:710–728.

[55] Freeman, W. J., B. C. Burke, and M. D. Holmes. 2003. Aperiodic Phase Re-Setting in Scalp EEG of Beta-Gamma Oscillations by State Transitions at Alpha-Theta Rates. *Human Brain Mapping* 19: 248–272.

[56] James, W. 1890. *The Principles of Psychology*. Cambridge, MA: Harvard University Press.

[57] Jackson, J. V. 1987. Idea for a Mind. *Siggart* Newsletter, 181:23–26.

[58] Negatu, A., T. L. McCauley, and S. Franklin. in review. Automatization for Software Agents.

[59] McAllester, D. A. and D. Rosenblitt. 1991. Systematic nonlinear planning. In *Proceedings of the Ninth National Conference on Artificial Intelligence* (AAAI-91), Vol. 2. 634–639. Anaheim, CA: AAAI Press.

[60] Minsky, M. 1985. *The Society of Mind*. New York: Simon and Schuster.

[61] Cox, M. T. 2005. Metacognition in computation: a selected research review. *Artificial Intelligence* 169:104–141.

[62] Zhang, Z., D. Dasgupta, and S. Franklin. 1998. Metacognition in Software Agents using Classifier Systems. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*. Madison, Wisconsin: MIT Press.

[63] Damasio, A. R. 1999. *The Feeling of What Happens*. New York: Harcourt Brace.

[64] Strawson, G. 1999. The self and the SESMET. In *Models of the Self*, ed. S. Gallagher, and J. Shear. Charlottesville, VA: Imprint Academic.

[65] Gallagher, S. 2000. Philosophical conceptions of the self: implications for cognitive science. *Trends in Cognitive Science* 4:14–21.

[66] Baars, B. J., T. Ramsoy, and S. Laureys. 2003. Brain, conscious experience and the observing self. *Trends Neurosci.* 26:671–675.

[67] Goldberg, I. I., M. Harel, and R. Malach. 2006. When the Brain Loses Its Self: Prefrontal Inactivation during Sensorimotor Processing. *Neuron* 50:329–339.

[68] Merker, B. 2005. The liabilities of mobility: A selection pressure for the transition to consciousness in animal evolution. *Consciousness and Cognition* 14:89–114.

[69] Johnston, V. S. 1999. *Why We Feel:The Science of Human Emotions*. Reading MA: Perseus Books.

[70] Franklin, S., and L. McCauley. 2004. Feelings and Emotions as Motivators and Learning Facilitators. In *Architectures for Modeling Emotion: Cross-Disciplinary Foundations, AAAI 2004 Spring Symposium Series*, vol. Technical Report SS-04-02. Stanford University, Palo Alto, California, USA: American Association for Artificial Intelligence; March 22–24, 2004.

[71] Franklin, S., and U. Ramamurthy. 2006. Motivations, Values and Emotions: 3 sides of the same coin. In *Proceedings of the Sixth International Workshop on Epigenetic Robotics*, vol. 128. Paris, France: Lund University Cognitive Studies.

Advances in Artificial General Intelligence: Concepts, Architectures and Algorithms
B. Goertzel and P. Wang (Eds.)
IOS Press, 2007
© 2007 The authors and IOS Press. All rights reserved.

A Working Hypothesis for General Intelligence

Eric Baum
Baum Research Enterprises
ebaum@fastmail.fm
http://whatisthought.com

Abstract. Humans can construct powerful mental programs for many domains never seen before. We address the questions of how this occurs, and how it could possibly be accomplished in software. Section one surveys a theory of natural understanding, as follows. One understands a domain when one has mental programs that can be executed to solve problems arising in the domain. Evolution created compact programs understanding domains posed by nature. According to an extrapolation of Occam's razor, a compact enough program solving enough problems drawn from a distribution can only be found if there is simple structure underlying the distribution and the program exploits this structure, in which case the program will generalize by solving most new problems drawn from the distribution. This picture has several important ramifications for attempts to develop Artificial General Intelligence (AGI), suggesting for example, that human intelligence is not in fact general, and that weak methods may not suffice to reproduce human abilities. Section 2 exemplifies this picture by discussing two particular thought processes, the mental program by which I solve levels in the game of Sokoban, and how I construct this mental program. A computer program under construction to implement my introspective picture of my mental program is based on a model of a particular mental module called Relevance Based Planning (RBP). Section 3 argues that programs to address new problems (such as my mental program to play Sokoban) can be constructed (both naturally and artificially) if and only if sufficient guidance is already present. It proposes a computational structure called a scaffold that guides rapid construction of understanding programs when confronted with new challenges.

Introduction

A striking phenomenon of human intelligence is that we understand problems, and can construct powerful solutions for problems we have never seen before. Section one surveys a theory under which one understands a problem when one has mental programs that can solve it and many naturally occurring variations. Such programs are suggested to arise through discovering a sufficiently concise program that works on a sufficiently large sample of naturally presented problems. By a proposed extrapolation of Occam's razor, such a concise effective program would only exist if the world posing the problems had an underlying structure that the program exploits to solve the problems, and in that case it will generalize to solve many new problems generated by the same world. It is further argued that the concise Occam program leading to human and natural understanding is largely embodied in the genome, which programs development of a modular program in the brain. We build upon this modular program in constructing further mental programs.

Section 2 exemplifies this picture by discussing two particular thought processes, the mental program by which I solve levels in the game of Sokoban, and how I construct this mental program. A computer program under construction (with Tom Schaul) to implement an introspective picture of my mental program is based on a new approach I call Relevance Based Planning (RBP). RBP is suggested as a model of a human planning module. By invoking domain dependent objects, it exploits the underlying structure of its particular domain, which in the case of Sokoban involves exploiting 2-dimensional structure. Several other domain dependent modules, intended to reproduce introspection, are under development.

Section 3 proposes a computational structure called a scaffold that provides guidance on how to rapidly construct understanding programs when confronted with new challenges.

A general contrast between the point of view of this paper and much work in AGI or Human Level AI, is that this paper is focused on the problem of how one can construct programs that understand new problems and problem domains. Much current work is based on systems that are expected to run existing programs, and either avoid facing the issue of automatically constructing new code, or assume it can be done by weak methods. The theory presented here argues that understanding is based on Occam programs, which are computationally expensive to find, because one must solve a hard computational problem to construct them. The necessary programs, for example in Sokoban, seem to be rather complex. Accordingly, I suggest that human intelligence is not really "general", but rather is powerful only at exploiting a class of problems arising in the natural world, and problems solvable with modules arising from these. New programs solving a new domain can be feasibly constructed only when one already has much of the computational structures necessary to build them, so that the search for the new program is tractable. This contrasts, for example, with the assumption that intelligence is based on weak methods-- methods not exploiting much knowledge [1]. Instead, I argue that understanding is based on having powerful domain dependent knowledge, and address the questions of how we can build such knowledge, what form it takes in humans and what form it should take in attempts at building human-level AI, and how it guides search for new programs.

1. Natural Intelligence

Turing argued compellingly that whatever is happening in the brain, it could be simulated in detail by a computer running the right software. Thus thoughts must be isomorphic to some particular computations. Turing's thesis gives us a precise language which we can use to discuss and model thought, the language of computer programs. This thesis, however, left us with some puzzles. A first important one is: what about this particular code causes it to *understand*? A second important one is: given that complexity theory has indicated that many computations are inherently time consuming, how does the mind work so amazingly fast?

Computational learning theory has explained generalization as arising from Occam's razor. The most studied context is concept learning, where one sees a series of classified examples, and desires to learn a function that will predict correctly whether new examples are examples of the concept or not. Roughly speaking, one can show that if one presents examples drawn from some process, and finds a simple enough function classifying most examples in a large data set, it will also correctly classify most new examples drawn from the process on which it hadn't been been specifically trained, it will generalize. Results along these lines from three viewpoints (VC dimension, Minimum Description Length, and Bayesian probability) are surveyed in chapter 4 of [2]. Yet another branch of related results can be found in the literature on universal algorithms, cf [3,4].

Such approaches start by formalizing the notion of what is meant by ``simple". One way is by program length, a shorter program (or alternatively, a shorter program that runs within a given time bound) is considered simpler.

A basic intuition behind these kinds of results is the following. If the process producing the data was not structured, the data would look random. No extremely simple program would then exist dealing correctly with the seen data. The fact that you are able to find a compact program correctly classifying massive data thus implies that the process providing the data actually had a compact structure, and the program was exploiting that structure to make predictions.

Such results have been known for decades, so why do we not yet have programs that classify concepts as well as people do? An answer is, we are unable to actually find a compact program (say a compact neural net) correctly classifying most interesting kinds of data, even for sets where people could classify. In fact, for a wide variety of sample problems, it has been proved NP-hard to find such compact programs, which indicates there can be no fast solution. The reader may be familiar with the Zip utility, in common use to compress computer files. Zip embodies a simple algorithm that runs rapidly, but is largely oblivious to the file being compressed. Generalization, by contrast, requires extracting the simple underlying structure particular to a given data set thus achieving a much smaller compression, and can not be done rapidly in such a generic fashion. It is only after the point where data is compressed beyond what is easy or generic that the underlying structure becomes apparent and meaningful generalization begins, precisely because that is the point where one must be sensitive to specific, surprisingly compact structure of the particular process producing the data. When such compression can be accomplished in practice, it is typically done by some algorithm such as back-propagation that does extensive computation, gradually discovering a function having a form that exploits structure in the process producing the data.

The literature also contains results that say, roughly speaking, the only way learning is possible is through Occam's razor. Such no-go theorems are never air tight -- there's a history of other no-go theorems being evaded by some alternative that escaped conception-- but the intuition seems reasonable. A learner can always build a complex hypothesis that explains any data, so unless the space of possible hypotheses is highly constrained by some inductive bias, there is no reason why any particular hypothesis should generalize. Note that the constraint does not have to be shortness of program-- in fact evolution seems to use a form of early stopping rather than just building the shortest hypothesis.

Human thought and understanding, of course, seem to encompass much deeper abilities than simple concept classification. However, the basic Occam intuition behind such classification results can be straightforwardly extrapolated to a conjecture about a general program interacting with a complex world: If you find a short enough program that rapidly solves enough problems, that can only be because it exploits compact structure underlying the world, and then it will continue to solve many later problems presented by the same world.

My working hypothesis is that this exploitation of structure is what understanding is. Humans are based on a compact program that exploits the underlying structure of the world. So, for example, we can solve new problems that we were not explicitly evolved to solve, such as discovering higher mathematics, playing chess or managing corporations, because underlying our thought there is a compact program that was trained to solve numerous problems arising in the world, so many that it could only solve these by exploiting underlying structure in the world, and it thus generalizes to solve new problems that arise.

How does a very concise program deal with many problems? Experience and a number of arguments suggest, by code reuse. It can be so concise by having a modular structure, with modules exploiting various kinds of structure that can be put together in various ways to solve various problems. As evolution produced a compact program to deal with the world, it discovered such modules.

This may explain why thought is so metaphorical. Metaphor occurs when we reuse a module associated with one context, to exploit related structure in another context. So when we speak of buying, wasting, spending, or investing our time, we are reusing a computational module or modules useful for valuable resource management. When we speak of buttressing the foundations of our arguments, we are reusing modules useful for building concrete structures. And so on.

Note that the kind of exploitation of structure involved here is rather different than what we usually think of in simple classification or prediction problems. If we simply find a concise enough program (for example, a small enough neural net) correctly classifying data points (for example saying whether images show a chair or don't), it will generalize to classify new data points (e.g. images) drawn from the same process. But simply finding a compact description of structure can be a separate problem from exploiting compact structure. In the Traveling Salesman Problem, for example, we are handed a concise description of the problem, but it is still computationally hard to find a very short tour. Roughly speaking, to find the shortest tour, we will have to search through a number of possibilities exponential in the size of the description. The claim is that the world has structure that can be exploited to rapidly solve problems which arise, and that underlying our thought processes are modules that accomplish this. And for many problems, we accomplish it very fast indeed.

Consider, for example, the integers. That the integers have a compact structure is evident from the fact that all of their properties are determined by 5 axioms-- but beyond this you know algorithms that you can use to rapidly solve many problems involving them (for example, to determine if a 50 digit number is even). My working hypothesis is that our mathematical abilities arise from Occam's razor. Roughly speaking, we have these abilities because there is a real a priori structure underlying mathematics, and evolution discovered modules that exploit it, for example modules that know how to exploit the structure of Euclidean 2 and 3 space. By evolving such modules we were able to solve problems important to evolution such as navigating around the jungle, but such modules perforce generalize to higher problems.

Mathematical reasoning is one example of an ability that has arisen this way, but of course the collection of modules we use to understand the world extends far beyond, as seen for example from the explanation of metaphors above. My working hypothesis is that each concept, basically each word in English, corresponds to a module (or at least, a method).

I expect that the mind looks like some incredibly elegantly written object-oriented code. That water is a liquid is presumably more or less represented by the class water having a superclass liquid, from which it inherits methods that allow you to mentally simulate it flowing, as well as solving various problems involving it. Experience indicates that such things are not readily coded into logic, especially not first order logic. Rather, I expect humans accomplish inference by using special procedures, that know how to do things like simulate liquid flows. My working hypothesis is that the brunt of inference by humans is not carried by modus ponens or some such general inference mechanism, it is in the various procedures that encode methods exploiting particular aspects of the structure of the world, such as the a priori structure of Euclidean 2 space or 3 space, the physics of liquids and hard objects and gases and so on. Thought is too fast to involve extensive searching over proofs, rather it is put together by executing powerful procedures encoded into a library. I expect words are more or less labels for code modules (roughly speaking nouns are classes and proper nouns instances of classes and verbs methods) so there may be tens of thousands of modules in the library. As we will discuss in Section 3, when we come to understand new problems, we do engage in search over programs, but it is a search so heavily biased and constrained by existing structure as to occur in feasible time.

Where in a human is this compact Occam program encoded? A number of arguments indicate that a critical kernel of it is encoded into the genome.

The genome is extraordinarily compact. Its functioning core (after the so-called "junk" is stripped away) is believed to be smaller than the source code for Microsoft Office. The brain, by contrast, is 100 million times bigger. Moreover the genome encodes, in a sense, the results of an extraordinary amount of data and computation: Warren Smith has recently improved on my estimate, and the best estimate now appears to be that some 10^{44} creatures have lived and died, each contributing in a small measure to evolution (footnote 95 in [5]).

The proposal of the genome as the Occam program is controversial, because many psychologists, neural network practitioners, and philosophers have argued that infants are born in a state of tabula rasa, a blank slate from which we acquire knowledge by learning. Learning theory, however, requires one have an inductive bias to learn. In my picture the genome encodes this inductive bias, programs that execute, interacting with sensory data, to learn. The learning so innately programmed appears quite automatic, reliably resulting in creatures with similar abilities provided that they are allowed interaction with the world during development.

Complexity theory suggests that learning is a hard problem, requiring vast computation to extract structure. Yet we learn so fast that we do not have time to do the requisite computation. This is possible only because creatures are preprogrammed to extract specific kinds of meaning. The bulk of the requisite computation, and thus the guts of the process from the point of view of complexity theory, went into the evolution of the genome.

Empirical evidence shows that creatures are in fact programmed with specific inductive biases. If a rat is shocked once at a specific point in its maze, it will avoid that corner. If a rat is sickened only once after eating a particular food, it will never eat that type of food again. However it is difficult to train a rat to avoid a location by making it sick or to avoid a type of food by shocking it. The rat is innately programmed to learn particular behaviors from particular stimuli. Similar results hold true for people. A wealth of evidence supports the view that children learn grammar rapidly and almost automatically, because of strong inductive bias built into their brain by their genome. Moreover, while humans can learn ``meaningful" facts from a single presentation, they would find it almost impossible to learn things they are not programmed to recognize as meaningful. This meaning is, in my picture, defined by underlying programs, largely coded into the genome, that exploit particular underlying structure in the world. Such programs, in fact, essentially define meaning.

Consider the development of visual cortex. The brain uses parallax to calculate the depth of objects. This calculation must be tuned to the width between the eyes. The DNA program is the same in every cell, but its execution differs from cell to cell as the chemical environment differs. As the brain develops, and the skull grows, the DNA program automatically adjusts brain wiring to compute depth. How did this evolve? DNA programs that better develop function in the chemical environment were preferentially selected. But the chemical environment in cortex includes neural firings stimulated by the sensory flow. Thus a program evolved that, in effect, learns the distance between the eyes. Similarly, the same DNA programs cells to develop into visual cortex or auditory cortex depending on the ambient environment of the developing cell. Similar mechanisms would explain development of more abstract thought, for example, Seay and Harlow's famous discovery that monkeys can only acquire normal social behavior during a critical period in development. Critical periods are a flag indicating genomic programming, as many aspects of development are carefully timed. The DNA program exploits the sensory stream (reflected in the chemical environment of developing cells) to grow appropriate neural structures, which may implement powerful procedures for reasoning about the world. Learning and development are two sides of the same coin, and so we should expect evolution of tailored learning programs.

If the genomic program encodes development of modules for a variety of meaningful concepts, everything from valuable resource management and understanding 2-topology to causal reasoning and grammar learning, this should be manifested through local variations of gene expression in developing brains. Gray et al. recently published an image in Science magazine that seemed to show quite detailed structure in gene expression [6]. As the technology improves, gene expression data should show programming of the kind required by this theory, or refute it.

The working hypothesis proposed here is that the genome encodes a compact ``Occam" program that (metaphorically) ``compiles", interacting with sense data, into executable modules we use to perceive and reason. The modules that are built may be relatively large and complex, yet still generalize because they are constrained by the compact genome. Lakoff and Johnson have cataloged so many diverse types of causal and temporal reasoning that they suggest there can exist no ``objective metaphysics" of time or causality [7]. I believe this is mistaken. Our universe does indeed display amazingly rich phenomena, but these all arise from an amazingly concise underlying physics. Our mental modules are varied, but constrained by a concise program to exploit this concise structure. Thus all the different types of causal reasoning are related and constrained.

The human mind evidently employs algorithms not explicitly coded in the genome. For example, the reader has procedures that allow him or her to read this text. Such knowledge is built as meaningful modules that invoke more basic modules. Experience suggests that complex programs must be built in such a modular fashion, sometimes called an abstraction hierarchy. The Occam's razor hypothesis suggests that the modules coded in the genome are meaningful precisely in the sense that powerful programs can be built on top of them. That is: these compact modules are such that in past experience, powerful programs have emerged to solve problems such as navigating in the jungle, therefore these modules should be such that powerful superstructures will emerge in new domains such as reasoning about higher mathematics.

Finding new meaningful modules is a hard computational problem, requiring substantial search. This suggests a mechanism by which human mental abilities differ so broadly from chimpanzees, who are genetically almost identical. Chimpanzees can discover new meaningful modules only over one lifetime of study. But humans, because of our ability to transmit programs through language, have cumulatively built and refined a vast library of powerful computational modules on top of what is coded in the genome.

This additional programming does not just include what we think of a technology, but rather includes qualities regarded as core mental abilities. For example, only humans are said to have a theory of mind in that we understand that other individuals may have different beliefs than our own, and reason accordingly. However, apes display behaviors indicating aspects of theory of mind; and plovers feign injury, limping off to distract a predator from their nest, only when the predator seems to notice the nest. Thus plovers attend to the perception of the predator and modify their behavior accordingly. This ability requires subroutines on which any theory of mind must rely. This suggests that the human theory of mind is a complex modular program built on top of meaningful modules coded more explicitly in the genome, and that humans have been able to discover this more powerful program over generations, because we pass partial progress on. Bedtime stories and fiction are means of passing such programs on to children. Psychophysics experiments are consistent with this view, showing that children acquire theory of mind over a period of years.

Thus this computational theory of mind, while it suggests that much of thought is coded in the genome, simultaneously suggests that most of the difference between human and ape cognition can be regarded as the product of better nurture. The key is that the program of mind is modular and hierarchic, and that the discovery of new modules is a hard computational problem. By virtue of our ability to communicate partial results, humans have made sustained progress in developing an ever more powerful superstructure on top of the innately coded programming.

In summary, computational learning theory has made a compelling case that generalization comes from Occam's razor. Understanding can be explained by a simple extrapolation of this as arising from evolution of a compact genomic program that builds a set of procedures that exploit the underlying structure. This and many other aspects of thought (for example, why the various aspects of consciousness, both subjective and objective, arise in this fashion), are explained in much more detail (and with many more citations) in [2], where some alternatives to the above working hypothesis are also briefly discussed.

This picture suggests that a truly ``general" intelligence is unattainable. Humans solve new problems very fast when we already have modules for dealing with them. Over hours or years, humans can solve somewhat more general new problems, those requiring construction of new code or new mental modules, only when we have modules that can be put together with only a reasonable amount of search (possibly guided or orchestrated by modules designed for putting other modules together into the right code). But according to the Occam thesis, all this machinery only works for problems in some sense drawn from a natural distribution of problems-- relating to the a priori structure of mathematics and the underlying compact structure of our universe. General problems (a likely example is given by arbitrary problems in the class NP) may simply not be rapidly solvable. Universal methods may be given that are argued in one sense or another to address such problems as well as possible [3, 4], but such solutions may take arbitrarily long and in my opinion don't relate directly to human intelligence; and may distract from what we want to study in AGI. To achieve AGI, I suggest, we will need to develop the kinds of modules on which human thought is built, that exploit the kind of underlying structure in our universe.

This picture explains why many AI programs do not seem to display ``understanding" [8]. These programs were not compact or constrained. Mostly, they were simply built by human programmers without Occam's razor in mind. Arguably, AI programs that have displayed some understanding have been built on very compact structures. For example, the core of chess programs such as Deep Blue, which do seem to understand chess quite well, is tiny: a program consisting of alpha-beta (a few lines of code) plus simple material as an evaluation function (also a tiny program) plus a short quiescence search routine would already play phenomenal chess if applied in an 11-ply search[1]. If we want to build an AGI, we will have to build something that

[1] Some readers may be suspicious of the assertion that Deep Blue in some sense understands chess since it does a huge search. A referee notes, for example, that a completely brute force exhaustive search program, a very few lines of code, would solve any finite problem eventually. In an attempt to capture normal notions of understanding, and yet discuss the subject in a concrete way, let us consider that an entity understands an item (or a domain) if it almost always does the right thing on the item or domain, and on almost all naturally occurring variations of it, and behaves in a reasonable way on other variations of it. By this definition, Deep Blue clearly understands most chess positions: pick almost any chess position you like, and it will make the right move. Moreover, Kasparov attempted to play toward types of positions not normally occurring in human play in a deliberate attempt to stump it, and largely failed. Moreover, you could vary the rules of chess within some reasonable range, and alpha-beta + material + quiescence would still play well-- if you vary far enough, you will have to change the evaluation function, but if you allow tweaks like that, the same methods extend strongly all the way in variation space to, for example, Othello. Thus this analysis technique provides a useful basis for a scaffold, that may be applied to understand various games, cf section 3. That is, a scaffold that learns an evaluation function and fits it into the alpha-beta + evaluation function + quiescence framework has so much bias, it could readily learn to process a wide variety of games effectively. In my view, this is not so dissimilar from how I might pick up a new game (although I would adjust and apply other scaffolds as well.) Similarly, the exhaustive search algorithm would indeed be said to understand finite problems, where one has the resources to perform the computation. (Its worth noting in passing that Deep Blue is very far from brute force, since it arrives at its determinations while examining an infinitesimal fraction of the search tree, indeed it examines a finite search tree yet would presumably be effective for many variations of chess with an infinite search tree.)

It's not hard to give positions which Deep Blue doesn't understand (cf (Baum 2004)for discussion). The simplest way is to consider variations where humans analyze the problem from a different angle that is impervious to scaling, and you consider the problem on an n by n board as n gets large. The same applies to the exhaustive search algorithm. So I am not claiming that Deep Blue's understanding is the same as ours -- we have many more modules and scaffolds in our minds that allow us better understanding.

It might be objected that time complexity should figure in understanding, and indeed it does. I am implicitly positing that the colloquial notion of understanding is relative to the amount of time alloted. Thus to be said to understand, the entity must do the right thing fast enough. For example, I might be said to understand some branch of mathematics if I could prove deep theorems in it, even though this might take me years, but you wouldn't say I understand it just because I might prove such a theorem if you gave me a millennium. Deep Blue, which as noted above is not brute force, is fast enough to work in tournament time limits, and in fact it's very far from apparent that Deep Blue does more computation than I do in understanding a chess position.

The main point I am making, however, is that generalization and thus understanding follows from an extrapolated Occam's razor. When you find a very constrained program that's effective on a large collection of examples, so constrained that it should not exist unless there is some very special structure underlying the examples, then it will generalize. If you find a program that is not so constrained yet works on the same set of examples, it can not be expected to generalize. The constraints here are a restriction on the representation space-- the program space must be constrained in the sense that it effectively can not represent all functions, so that it is unlikely that a solution to a random problem falls in the space. Thus lookup tables of size comparable to the number of examples, although they are blindingly fast, are not constrained and will reflects the Occam property.

It is possible that we will never have the computational resources to accomplish this. Finding Occam programs seems to generically be NP-hard at least, and thus may require extensive computational effort. A human may be no more able to sit down and write Occam programs exploiting the structure of the world in the way the mind does, than a human is able by hand to solve other NP-hard problems. Evolution had more computational resources for solving this problem than we ever will. On the other hand, we start with a big advantage-- our brains. The most promising approach, as I see it, is thus to build the AGI by a collaboration of human programmer and computer. The humans must encode as much inductive bias as possible, and extensive computation must build on this to construct Occam programs.

One problem with this Occam picture is that it doesn't necessarily tell us what the Occam code looks like, beyond being very compressed and produced by evolution. In fact, it suggests that there is no much better understanding of the Occam code than the code itself, since understanding comes from a compressed description, and the Occam code is already so highly compressed that further compression, if it even exists, should be very difficult to find. Moreover, according to the Occam intuition (and some of the formal results on which it is based), any very highly compressed program effective on the data seen, rather than only the most compressed possible program, is sufficient for generalization. NP-hard problems such as that of finding such compressed descriptions often have large numbers of local optima, which may look unlike one another in detail. For example, for any huge planar Traveling Salesman Problem, almost any planar tour will be quite short but such tours may share very few links[2]. This explains why the inner workings of trained neural nets are sometimes inscrutable. On the other hand, in my working hypothesis, the Occam core is in the genome, and the program in the brain is rather larger, thus admitting of a shorter description, so we might expect to be able to say something about the code in the brain. In the next section, I discuss a simple but highly non-trivial example of thought, attempting to further illuminate the nature of the code and how it is produced.

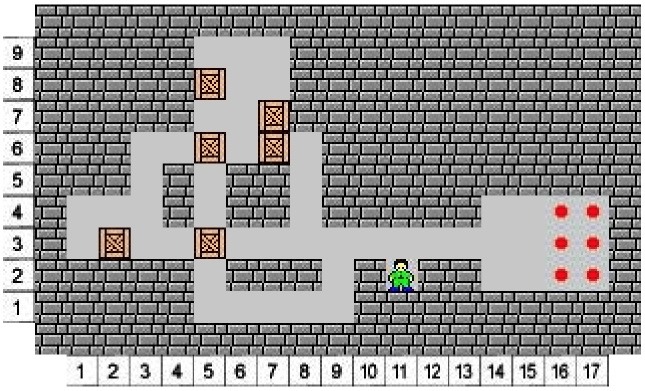
2. Introspective Sokoban

Sokoban is a single player game consisting of a collection of two dimensional planning puzzles called levels. The player controls a robot that can push barrels around a maze (see figure). Only one barrel can be pushed at a time, and the object is to push all the barrels into marked goals. The game is best understood by playing a few levels-- numerous levels can be found on the web cf: [9,10]. Sokoban is p-space complete [11]

not generalize, and many AI programs are akin to this. However, I do not restrict to code length as the only possible constraint. It is entirely possible that one might achieve generalization with constraints that combine a time restriction and coding restrictions, and indeed I implicitly assume throughout that the Occam program solves the training examples fast enough. As discussed in chapter 6 of [2], evolution actually seems to have employed as constraints a combination of early stopping, evolvability (which strongly affects what representations are achievable), code length, and time complexity.

[2] Boese [12] found empirically that most short tours found by local search algorithms are related. However, there are an exponential number of locally optimal (for example, planar) tours and one could easily construct examples at various edit distances. Even in relaxation systems, such as metals or crystals, where it is trivial to compute the global optimum, physical relaxation processes produce domain structures, and two different configurations of domains will have substantial hamming distance.

so there is no hope of producing a (fast) general solver[3].



Humans have however crafted a large collection of ingenious levels that humans find interesting to solve. One reason the domain is interesting for students of AGI, in fact, is that these levels appear to have been crafted so as to exercise a range of human cognitive abilities.

From the point of view of understanding general intelligence, we are interested in two problems. After having practiced the game for a while, I have a toolbox of techniques-- a mental program involving several distinct modules-- that I apply to solve new levels. The first problem is, what does this mature program look like? The second problem is, how do I build this program when I first begin playing Sokoban? The key problem in both cases is time complexity. Searching over all possible solutions, and all possible short programs for solving Sokoban, is way too slow. So we must have vast knowledge that we bring and use. Since all our thoughts correspond to computations, my working hypothesis is that this knowledge is in the form of some procedural program like the kind described in the first section that allows us rapidly to construct an effective solving program. We come to the game with numerous modules useful for

[3] Roughly speaking, the standard proof that a particular problem class C is hard is given by showing that you can map another hard problem class H into it. Since there is (believed to be) insufficient underlying structure to solve H rapidly, this then exhibits a subclass of C that is not (believed to be) rapidly solvable. In the case of Sokoban, the proof proceeds by constructing, for any given Turing machine with given finite tape length, a particular Sokoban instance that is solvable if and only if the Turing machine halts in an accepting state. This leaves (at least) two outs for solving problems in C rapidly. First, since the mapping is into, with range only a small subset of instances of C, almost all instances of C may be rapidly solvable. For example, in the case of Sokoban, it intuitively seems that random instances can only be hard if they have a high enough density of barrels and goals, because otherwise solving the different goals decouples. Furthermore, it is plausible that random solvable instances with too high a density of barrels and goals would be over-constrained and hence readily solvable by algorithms that exploit over-constrainedness, which I believe humans often do, including in Sokoban. It would be an interesting experiment to write a random instance generator and see in what domains humans can easily solve large random Sokoban instances. Second, such mappings generically map a problem instance in H of size S into a problem instance in C of size Sk, for k some constant of rough order 100. Thus an option for solving instances in C that derive from instances in H is to perform the inverse mapping, and apply an exponential time algorithm for solving instances in H. This will be practicable for substantial (although of course, not asymptotic) instances of C. Human strategies for Sokoban, roughly speaking, exploit both these avenues. They attempt local solutions, which will work if the density of constraints is low and the problem factors. They also propagate constraints arising from barrel interactions causally, which seems likely to solve over-constrained random problems, and also ferrets out mapping structure.

problems such as planning in 2 dimensions, for example modules that calculate aspects of 2-topology like the inside of an enclosure, and modules that perform goal-directed planning. Using this head-start, we rapidly construct (within minutes or weeks) programs for analyzing Sokoban and other problems we encounter.

Introspection into my mature thought processes in Sokoban-solving motivated the design of a general planning approach that I call Relevance Based Planning (RBP). RBP may be viewed as a domain independent core algorithm that organizes computation using a number of domain dependent objects. RBP employs modules that specify various domain dependent knowledge, including all or some of the following: identifying which classes of objects in the domain are potentially affectable by operators, specifying methods for dealing with classes of obstacles, specifying contra-factual knowledge about what states would be reached by applying not-currently-enabled operators in a given state if affectable obstacles to applying them could be dealt with, identifying local configurations of objects that prevent solution of goals (comprise a deadlock), and, where multiple potentially interacting goals are to be achieved, analyzing constraints on the order in which goals can be achieved.

RBP first applies a method to find candidate plans that could potentially succeed in solving the problem if various affectable obstacles could be dealt with. An example of such a candidate plan might be a plan to get from point A to point B, which has to get through two locked doors, but otherwise does not need to do anything impossible, such as teleport through a wall. A powerful way to generate such candidate plans is by dynamic programming, invoking the contra-factual knowledge (eg to say, if only I could unlock this door, I could get to the other side, and doors are known to be potentially unlockable, so I will proceed to see if there is a path from the other side to my goal). RBP then refines such candidate plans to see whether the obstacles can be dealt with. By proceeding in time order, it deals in this refining only with states it knows it can reach (thus avoiding much irrelevant work often engaged in by planners). If it hits a deadlock or other obstruction, it employs domain-specific knowledge for dealing with it (e.g. applies a module for dealing with that particular obstacle class, such as a module that searches for a key and uses it to unlock a door, or, because it knows what elements of the domain are comprising the deadlock, tries to move them out of the way in ways that would prevent the deadlock from arising). Also, as it takes actions on the domain simulation, it marks affected objects. If an affected object later obstructs a desired action, it considers the alternative plan of doing the other action first, to see if it can escape the obstruction. It contains a procedure for ordering the search over possible candidate plans in an optimized fashion and a hash table over states to avoid duplicating work. The algorithm that results is called *Relevance* Based Planning because it searches only possible actions that are directly relevant to achieving the goal, that is: are refining a candidate plan already known capable of achieving the goal if certain obstacles known to be potentially affectable are dealt with, and furthermore are actions that actually affect a relevant obstacle or are relevant to a plan to affect a relevant obstacle.

Tom Schaul and I first tested RBP on the sub-problem within Sokoban of getting the pusher from point A to point B without creating a deadlock. The domain dependent knowledge necessary for solving this task is relatively straightforward. Barrels can be pushed, walls can not, if the pusher attempts to move somewhere and a barrel is in the way, then the pusher may arrive in the position on the other side in the event that the barrel can be dealt with, and so on. RBP solves this problem in a rapid, efficient and very natural way. Introspection does not suggest any important differences between

RBP and my mental planning process on this problem[4].

Schaul and I have since been extending this approach to tackle the general problem of solving Sokoban instances. For this purpose a number of modules have been created. These modules implement concepts that I believe I use in solving Sokoban, and thus illustrate the procedural nature of human reasoning.

One is a deadlock detector capable of recognizing deadlocks arising from the inability to get behind any barrier formed by barrels. If a deadlock is found, the detector returns a list of the barrels comprising the deadlock (so that RBP can consider pushes of these barrels relevant to preventing the deadlock from arising.) The concepts of "deadlock", "behind", and "barrier" are fundamental to human thought on Sokoban, and seem naturally to require procedures to implement or recognize them.

Note that barriers are non-local (they can involve widely dispersed barrels that nonetheless form a barrier separating one region of the level from another). RBP straightforwardly follows complex causal chains of reasoning where it will attempt to move a barrel involved in a barrier on the far side of the level in the effort to prevent a deadlock arising when a barrel is pushed on the near side, while ignoring barrels that are not causally involved.

Another module analyzes goal placement and returns orders in which the goals can be solved. If the goals are solved in an invalid order, an earlier solved goal may prevent later solution. Some orders may require that regions be cleared before a goal is solved, or that barrels be stowed in particular locations before a goal is solved so that they will be in position to later be moved into goals. The goal area analyzer calculates all such constraints by finding a hypothetical barrel configuration from which the problem could be solved in the given order. Note that the goal area analyzer, since it doesn't pay attention to the actual locations of barrels in the domain, can return orders that are not in fact achievable from the initial position of the Sokoban level, if the reason they are not achievable is because of constraints to which it is not sensitive, such as if current positions of barrels block pusher access which is necessary for achieving a given order. We have thus factored the problem cognitively into the calculation of a valid goal order, and the calculation of how to push the barrels into that order, or if the order is not achievable, finding a different valid order. Introspectively, I use a related factorization.

Another module finds a bipartite match of all barrels to goals, where each barrel matched to a goal is capable of being pushed to the goal if one disregards other barrels that may be in the way. This is built on top of a "flowmap" that amounts to a perceptual analysis of the Sokoban level. The flowmap is a concise data structure that contains a graph of all possible pushes (under the assumption that all other barrels are deleted), and also indicates the strongly connected components of the level. The matching

[4] All high level decisions are made by RBP in such a way that only a very restricted set of alternatives known to be relevant are considered. As finer possibilities are encountered, RBP may run out of knowledge that restricts what possible moves may be relevant. Then, if it wishes to be complete, it must search over all remaining possible moves. For some problems, one could imagine improving it by giving it other or better procedures that rule out some of these as irrelevant. In the problem of moving the pusher from point a to point b, if some barrel c obstructs a desired push, RBP might consider all pushes of c as relevant, whereas a human might say that some of them should be ruled out as not possibly useful. But the human has to do some computation to decide that this push is not useful. Some of such computations are done, both according to my introspection and in our program, by perceptual analysis, and others are done at the level of the planner. I wouldn't assert that the division is identical between my mind and our program, only that I can't point to an obvious difference and don't expect any differences on this problem are critical-- for example that they would lead to time-complexity scaling differently as more complex problems are considered.

analyzer thus reflects several constraints on solution (such as the concept of 1-1 matching). I am not conscious of mentally working out an exact assignment before starting in on every Sokoban instance-- but I am conscious of thinking about it whenever the instance is such that it may be problematic, indicating I am attending to it.

Another module finds and implements forced moves. In Sokoban, it frequently happens that once one is in a given position (very often the starting position has this nature, but it also arises during search) a sequence of pushes is forced (up to transpositions), because any other pushes will lead to deadlock. When such a circumstance is found, it makes sense to immediately make this sequence of pushes rather than analyzing the current position further. Again, introspection indicates that I interleave state space search in this way with higher level cognitive analysis such as that computed by the other modules and the overall planning algorithm.

The overall Sokoban program works by applying Relevance Based Planning to the problem of solving a given Sokoban level. It first finds candidate plans that could solve the level if affectable obstacles (namely, the other barrels) could be dealt with. Such a candidate plan combines a valid goal order and a valid matching. It then refines such plans. This involves a backtracking search where, when a given plan is ruled out, it backs up to a previous branch that is relevant to solve the particular obstacle preventing the previous plan from succeeding. It explores only backups and changes that are relevant to fixing a flaw in a previously considered plan.

Schaul has implemented and largely debugged a first cut at all of the above modules, but some components are not yet implemented efficiently or effectively enough, and several have not yet been integrated. At present, our program is solving 31 of the 90 standard levels(Meyers).

A few comments on the effort as it stands. First, it is clear that I do calculate the above modules. Some researchers are suspicious of introspection, but it is nonetheless objectively verifiable that I calculate these concepts, because I can answer questions such as: "what barrels in this configuration form a deadlock". Unfortunately, the means I use to calculate them do not seem introspectively to be the same as how Tom has coded them, and this is causing difficulties-- his code sometimes has problems in calculating some one of these modules that I find trivial. Nonetheless, I do compute similar concepts and integrate them in related (but possibly more sophisticated) ways. My mature mental program has thus got to be quite complex. I must have some ability to build this complex program on a timescale of days. I would assert that any system aspiring to do human-level AI will need the capability to build procedures of this general nature and complexity.

Second, it seems at best borderline possible to implement my mental program for Sokoban by hand. Doing so has so far required a few man years of work, and is not yet finished. Even if it can be accomplished for Sokoban, one might imagine that other domains that are less well formalized would prove daunting. There are also a few computational modules that I introspectively use in Sokoban that have not yet been coded, and may prove beyond our abilities, or at least beyond our funding. It's not yet clear whether these are essential, or whether instances I solve using them can all be solved roughly as easily using other implemented techniques. Some of them certainly improve efficiency. For example, I mentally cache meaningful computations in a way our system does not yet capture. It's also plausible that if I encountered a new level that required a new kind of technique, I would build it mentally, so simply implementing a program that solves any given set of levels might not exhaust human capability.

But perhaps the biggest problem is, our ability to introspect about how modules work seems limited. The top level RBP algorithm seems much more accessible to introspection than the inner workings of the modules it calls. This is consistent with the picture of consciousness put forth in chapter 14 of [2], where awareness is identified with computations of a top level module that accesses lower level modules only through procedure calls and the like, and is thus not able to directly examine their mechanism. As a result, the algorithms implemented within the Sokoban modules are not necessarily similar to the algorithms implementing similar computations within human thought, and it has turned out as a result that they do not always perform adequately. It appears that such modules will have to be learned, rather than hand programmed.

3. Code Construction and Scaffolds

Two key question are thus how do I mentally build an algorithm to play Sokoban when I take up the game, and how can we produce a program that is capable of building such programs when presented new problem domains?

My working hypothesis is that building such a program as fast as I do is only possible if almost all pieces of the program are already constructed, and only a series of relatively small searches is necessary to put them together. So I suggest that I have mentally already something very much like an RBP module, that needs only to build the various domain dependent objects in order to apply to Sokoban. And moreover, I have various modules already present that are useful for building these modules. I am then able to build it by a series of module constructions, each involving sufficiently small search that it is feasible.

Experience indicates that evolutionary programming is quite capable of producing programs that solve interesting problem domains, if it is given sufficient guidance, for example in the form of an instruction set containing pertinent macro-instructions, and a series of increasingly hard problems that allow it to make each new programmatic discovery in a feasible length of time. For example, Igor Durdanovic and I performed experiments with an evolutionary program we called Hayek [13, 14, 2]. Hayek received at least three kinds of guidance. First, it was based on an evolutionary economic system designed to bias in the building of a modular program, and to rapidly prune candidate modules not working in consort with the rest of the system. Second, it was presented problems with considerable structure, and given training examples that increased in size and difficulty as it learned. Third, it was given some guidance in the form of its instruction set, which in some runs included useful macros. Given all this, it was able to rapidly produce programs that solved interesting problems. Given a useful macro, it rapidly generated a program that solved very large Blocks World Problems. Given an expression language that allowed useful pattern matching and turned out to admit a 5 line program solving arbitrary Blocks World instances, it generated such a program-- but only when given syntax that restricted the search space by building in some aspects of the topology. Not given such syntax, it failed for a simple and obvious reason-- it couldn't in a week of computation find a program that solved even the easiest problems presented, and thus it got no feedback whatsoever, hence was engaged in blind search. Control experiments using the same instruction sets, but dropping particular aspects of the economic system, showed that the economic system also provided helpful bias useful to the successful outcome.

Likewise, as I read Schmidhuber's description of his experiments building a Tower of Hanoi solver [4], what jumps out is that after putting in a few carefully chosen macro-instructions, and by training on increasingly larger instances, his system was able to find a powerful program after affordable search[5].

We also tested such a hypothesis in preliminary experiments in Sokoban. Given instructions such as ``getBarrelTo(B,L)" (which computed and applied a sequence of pushes taking barrel B to location L) and "openupaZone" (which computed and applied a sequence of pushes getting behind a nearby barrier), Hayek was rapidly able to create a solver capable of solving an interesting Sokoban level [15].

What does introspection tell me about how I built my mental program to play Sokoban? To begin with, when I approached Sokoban, I already had in my mind at the least something much like RBP implementation of move(A,B) (the module that takes the pusher from point A to point B). I use this for navigation constantly in real life-- calculating how to get from my seat to go get a drink, for example, even if some chairs are in the way. This module also (with minor amendment) allows me to push a barrel from point A to point B[6]. Note also that animals need to navigate around the world. Although move(a,b) involves somewhat complex code, it seems likely evolution discovered something like it and built it in to the genome. After all, no-one doubts that evolution discovered and built in the program for the ribosome, which is rather more complex.

I started doing pushing barrels toward goals,, and shortly encountered the possibility that filling one goal would interfere with later goals. I then recognized that the problem separated into bringing barrels up and finding a non-obstructive goal order, and built a goal order analyzer. It's not entirely obvious how I recognized here that the problem factored-- my working hypothesis is that I essentially already knew it, because I had knowledge coded in that I could analyze locally in space, and locally in time; that I could figure how to bring an object into a space and then separately analyze how to use it there. I expect such knowledge is already coded into animal genomes. And as I continued to address Sokoban levels, I encountered several other conceptual problems, each of which I recognized as separately solvable; in each case, I suspect, recognizing this because I already essentially had a module coding for it in another context, or a programmatic structure that told me it could be analyzed separately[7].

---

[5] Schmidhuber's paper emphasizes the bias-optimality of his program search technique. The extent to which ``bias-optimality" was helpful was not empirically tested by comparison to a hill-climbing alternative. Roughly speaking, Bias-optimality was defined to mean, a search over possibilities for the shortest solution, where the search never invests a much higher fraction of one's computation in a particular candidate than the likelihood that it is the best program according to one's bias. Given no particular bias, one thus searches uniformly breadth first. By contrast, hill-climbing roughly speaking adopts the position that good programs are clustered. If one wants to find the absolute shortest program, making variations in a good program might be a worse idea than searching in an unbiased way. But in a domain where good programs are clustered, the bias-optimal approach might fail where a hill climbing approach would find a program sufficiently compact to generalize. For example, Ellington and Szostack [16] suggested that RNA might find interesting programs because the space of RNA sequences contains a sufficient density of sequences that have a small ability to catalyze a particular reaction, that one such sequence can be found with feasible-size random search; and then hill-climbing on this sequence may rapidly generate a good catalyst. It won't be the best possible, but an unbiased search for the best would likely be completely infeasible.

[6] I also had previously developed the ability to apply this module in navigating around a map, from a top down view. I recall watching my young child develop this ability on a video game, and it took some days or weeks to construct.

[7] My ability to continue adding new techniques when faced with new problems requiring them, is thus conjectured to rest on my large existing library of modules not yet tapped.

My working hypothesis is thus that code may be feasibly constructed if enough guidance is provided, and we should ask how and in what forms guidance is provided within humans, and how we can provide guidance for artificial systems.

My model for how pre-coded structure can guide program construction is based on what I call a ``scaffold". A scaffold is a procedure, which may have arguments, together with annotations that give guidance in how to construct or select procedures to feed in to the arguments. Thus a scaffold may be written:

$P(a_1,a_2,...,a_n)[c_1,c_2,...,c_n]$

where P is a procedure or a function, the a_i are its arguments, and the c_i are annotations, c_j giving instructions in how to construct the j-th argument. Here $P(a_1,a_2,...,a_n)$ may be a specific procedure or function in the ordinary sense of computer programming, and may be written in a standard programming language, such as C, Python, or Lisp. P may also take an indefinite number of arguments, as for example dotted tail notation supports in Lisp.

When a scaffold is used, it first must be trained. In this phase, programs are constructed or evolved that substitute for the arguments. The annotations guide this process. The trained scaffold is then a module that can be applied.

An example of a scaffold without annotations might be the following. In Schmidhuber's experiments, he supplied some key macro-instructions called defnp, calltp, and endnp, each a separate module about 15 primitive instructions long. These proved to be useful for constructing recursive functions, and after he added them to the instruction set, his searcher was able to discover a program solving Towers of Hanoi after some days of search. But in fact, the final program's use of these instructions turned out to be in the forms (defnp a calltp b endnp) and (defnp a calltp b calltp endnp) where a and b represent slots into which other code was substituted. So maybe what was really wanted was a framework (defnp ((a calltp) b .) endnp) where by the notation (x .) I mean an arbitrary number of copies of structure x (biased toward a small number). Given such a structure, we have to construct code for a number of modules (in this case, by use of the dot notation, an indeterminate number of modules but in many cases a fixed number), and substitute them in to give the final module. This would have restricted the search space for the overall program much more than the simple instructions that Schmidhuber in fact used, which for example did not contain ordering information. The idea of scaffolds is to support such constructions, and to furthermore greatly guide the search for appropriate programs by allowing appropriate annotations.

There are at least five types of annotations that may be supported. Type 1 annotations suggest plugging in another scaffold in place of a particular argument. A list may be supplied, and tried in order (until one is found that works). These scaffolds are trained also (in depth first manner, discussed more below).

Type 2 annotations are text, giving instructions for example to a human programmer. (These same kinds of instructions could be read and processed by an appropriate module or program, provided it had sufficient knowledge already coded in to recognize and appropriately respond. At the present state of AGI development, human programmers are more readily available.) They often suggest supplying certain kinds of training examples to learn a program. For example, a type 2 annotation might suggest supplying examples of deadlocks to a module constructor to produce a deadlock detector, which would then be substituted in to the appropriate place in an

RBP scaffold with a slot demanding a deadlock detector. My hypothesis is, whenever we can separate out a sub-problem in a useful way, and either supply code for it, or train it from independent examples, that greatly cuts the search in finding the overall program, so such sub-problems should be addressed first where feasible.

Type 3 annotations specify particular kinds of module constructors to be used in constructing a program for a slot. By a module constructor, I mean a program that is fed such things as training examples, an instruction set out of which to build programs, an objective function, etc., and which then builds an appropriate program, for example by performing search over programs or evolutionary programming. For example, I have developed specific types of evolutionary economic systems (EES) that evolve programs that efficiently search for solutions, and variants of these that work in adversarial environments such as games. Standard evolutionary module constructors evolve programs to solve a problem. Hayek, for example, discovers a collection of agents so that, presented with a problem, a single sequence of agents will be considered that (hopefully) solves it. These new types of EES evolve programs that apply a focused search when solving a new problem. They thus learn search control knowledge appropriate for responding to various outcomes. Certain classes of problems seem to require search, so they will be appropriate in certain slots of scaffolds. In other slots of scaffolds, a neural net training might be more appropriate.

Type 4 annotations specify or constrain which instruction sets or instructions are to be used in building the program. Program evolution or construction is much more efficient when it is restricted to building the module out of instructions likely to be appropriate for the task at hand, and as few distracting instructions as possible. Guiding the instruction set can make a vast difference to the efficiency of a module constructions.

Type 5 annotations specify improvement operators (e.g. kinds of mutations to be used in evolving a module) (which again may be critical). Other kinds of annotations might be imagined.

Scaffolds may be trained by a straightforward algorithm. We begin by walking down depth first filling in scaffolds suggested by annotations of type 1 for particular slots. When we find an internal problem with independent training data (a sub-problem that can be independently trained, for example with an annotation specifying what data is to be supplied) we train that first, recursively, and assuming we find code that solves that sub-problem, we fix that code. Alternatively if we fail on such a specified sub-problem, we backup to a previous choice point in the search. When doing a module construction (for example, at the top level, with candidate scaffolds within filled-in and all internal independent problems already solved) we use a method that preserves whatever parts of the structure have been fixed. For example, we may do evolutionary programming where the only crossovers and mutations that are allowed respect the fixed parts of the code; or we may do search where parts of the code are fixed.

Note that this algorithm involves a certain amount of search- but if the scaffolds are appropriately coded, vastly less than would be necessary without them.

Of course, especially if annotations indicate what is wanted, a programmer may supply some modules by hand. This may sound contrary to the spirit of AGI, but actually much of human thought works that way also. In my picture, for example, humans approach games with (among other things) a scaffold that knows about search and needs to be fed an evaluation function. But when you first learned to play chess, you did not develop your own evaluation function. A teacher told you, consider material, 9 points for a queen, 5 for each rook, and add it all up. Later, you improved this evaluation function with other terms.

Some examples of interesting scaffolds are the following. A scaffold for combining causes. It asks to be presented with training examples of cause 1, and then applies a specified EES module constructor to build a collection of agents dealing with cause 1. Then it repeats this process for as many different causes as are presented. Finally, it extracts all the separate agents, merges them into a single EES, and trains this on presented examples in order to create code for interaction effects between the causes. A particular application for such a scaffold might be to solving life or death problems in Go. Game search Evolutionary Economic Systems that have agents suggesting moves killing or saving groups are first trained on examples with a single theme-- for example problems where the group can run to safety (or be cut and so prevented from running to safety). A second such G-S EES is trained on examples where the group can make internal eyes, or be prevented from making internal eyes. And so on. The agents are then extracted from these individual EES-es, and placed into a single EES, which is then trained on general examples (and thus learns to deal with interaction effects, for example feinting at a run in order to put down a stone useful for making internal eyes.) Such a scaffold includes fixed code: such as the EES code that holds auctions and determines what computation is done given a particular collection of agents reacting to a particular problem. It also includes annotations, specifying that a sequence of examples be presented of differing causes; specifying that specific module constructors are applied to learn from them; possibly specifying that specific instruction sets (e.g. pattern languages) are used to code the agents. It breaks up an evolutionary programming problem that would possibly be intractable into a series of more tractable ones.

A scaffold for graph based reasoning GBR(W,P,Q,R)[annotations] embodies knowledge of two dimensional graphs. It is presented with W, a world state supplied to the program consisting of a grid (representing a 2-dimensional problem) with a data structure assigned to each node of the grid (representing state at that location). The annotation specifies that an appropriate W be presented. P is a program that processes W and marks grid points according to whether they belong to local structures, which we call groups. The annotation specifies that either such a P should be supplied, or examples supplied from which it can be evolved. Q then is a program that evaluates groups, in the context of neighboring groups. Again, Q may be supplied or learned from examples. Finally R is a program that combines Q evaluations to an overall evaluation. R may be chosen from a list of useful scaffolds and functions (summing values will often be useful, in some adversarial games a complicated alternating sum will be appropriate) or learned from examples.

RBP can be formulated as a scaffold. It contains a fixed procedure that finds candidate plans, then organizes the refinement of candidate plans to find an effective plan. To find the candidate plans, it needs to call domain dependent operators that must be supplied or learned. The fixed procedure may code in dynamic programming, but it must plug in actions and a simulation that are domain dependent, and in order to do this it must have a module that recognizes which obstacles to applying operators may be affected, and specifies what state will be reached by contra-factual application of an operator (under the hypothesis that one or more affectable obstacles can be overcome). It also benefits from a supplied, domain dependent deadlock detector, which not only detects configurations of objects preventing success, but identifies the objects. But such modules as a deadlock detector, or a contra-factual state generator, can in principle be learned from supplied examples, and if appropriate examples can be supplied by a programmer (or generated automatically), this is a much smaller task than producing a program to solve the initial planning domain at one fell swoop.

Conclusion

My current working hypothesis is that understanding a domain requires having a library of scaffolds and modules exploiting the structure of the domain, either allowing problems to be immediately solved, or giving sufficient guidance that programs solving them can constructed with feasible amounts of computation. Such structure was evolved before humans came on the scene, and humans have built extensively on it, greatly increasing the power of human understanding over ape. Evolution threw vastly more computational power at this problem than we have available, and the human contribution involved the efforts of billions of brains, so constructing understanding systems may be difficult.

However, we have advantages that evolution didn't have-- namely our brains, which use all this previous structure. It is thus possible we can construct the requisite structures, and more likely still that we can get there by a collaboration of programming and computation. So the path I propose is that we set out to construct useful modules and scaffolds. When we come to a module embodying a concept that we can provide examples of, but not readily program, we attempt to produce it using a module constructor. If that is too difficult, we may supply useful sub-concepts, or recursively produce the useful sub-concepts from supplied examples and instructions. At any given step, both at top or at bottom, we have available automatic module construction as well as programming. That is, a module embodying a new concept can always call existing programs if their names are supplied in the instruction set to the module constructor; and if it can not readily be built like that, we may first try to produce (by training or programming) other useful sub-concepts. The only requirement is that enough guidance be provided, either by existing scaffolds, or by programmer intervention, or by supplying training examples, that each module construction be feasible.

Acknowledgements

Work funded in part by DARPA under grant #HR0011-05-1-0045 Thanks to Tom Schaul for writing the Sokoban program.

References

[1] Laird, J.E., A. Newell & P.S. Rosenbloom.(1987) SOAR: An Architecture for General Intelligence. Artificial Intelligence 33:1-64,.

[2] Baum, Eric. (2004) What is Thought? MIT Press, Cambridge, MA.

[3] Hutter, M. (2006) Universal Algorithmic Intelligence: A Mathematical Top->Down Approach,pp 228-291 in Artificial General Intelligence (Cognitive Technologies) (Hardcover) by Ben Goertzel and Cassio Pennachin (eds), Springer

[4] Schmidhuber, J.(2004) Optimal Ordered Problem Solver, Machine Learning 54, 211-254.

[5] Smith, W. D. (2006) Mathematical Definition of `Intelligence' (and Consequences), preprint, reference 93 on http://math.temple.edu/~wds/homepage/works.html

[6] Gray, P A. et al,(2004) Mouse Brain Organization Revealed Through Direct Genome-Scale TF Expression nalysis Science 24 December 2004: Vol. 306. no. 5705, pp. 2255 - 2257

[7] Lakoff, G., Johnson,(1999) M. Philosophy in the Flesh, the embodied mind and its challenge to western thought, New York, Basic Books.

[8] Dreyfus, H. L.,(1972) What Computers Can't Do, Cambridge MA MIT Press

[9] Myers, A. Xsokoban, http://www.cs.cornell.edu/andru/xsokoban.html

[10] Nugroho, R. P., (1999), Java Sokoban, http://www.billybear4kids.com/games/online/sokoban/Sokoban.htm

[11] Culbertson, J. C., (1997) Sokoban is PSPACE-complete. Technical Report TR 97-02, Dept. of Computing Science, University of Alberta.

[12] Boese, K. D. (1995) Cost Versus Distance in Traveling Salesman Problem, UCLA TR 950018, http://citeseer.ist.psu.edu/boese95cost.html

[13] Baum, E. B. & I. Durdanovic. (2000) Evolution of Cooperative Problem-Solving in an Artificial Economy. Neural Computation 12:2743-2775.

[14] Baum, E. B. & I. Durdanovic.(2002) An artificial economy of Post production systems. In Advances in Learning Classifier Systems, P.L. Lanzi, W. Stoltzmann & S.M. Wilson (eds.), Berlin: Springer-Verlag, 3-21.

[15] Schaul, T. (2005) Evolution of a compact Sokoban solver, Master Thesis,, École Polytechnique Fédérale de Lausanne, posted on http://whatisthought.com/eric.html

[16] Ellington, A. D., and J. W. Szostack, (1990), In vitro selection of RNA molecules that bind specific ligands. Nature 346:818-822.

Advances in Artificial General Intelligence: Concepts, Architectures and Algorithms
B. Goertzel and P. Wang (Eds.)
IOS Press, 2007

© 2007 The authors and IOS Press. All rights reserved.

From NARS to a Thinking Machine

Pei WANG
Temple University, Philadelphia, USA
http://www.cis.temple.edu/~pwang/

Abstract. NARS is an AGI project developed in the framework of reasoning system, and it adapts to its environment with insufficient knowledge and resources. The development of NARS takes an incremental approach, by extending the formal model stage by stage. The system, when finished, can be further augmented in several directions.

Keywords. General-purpose system, reasoning, intelligence, development plan

1. Introduction

NARS (Non-Axiomatic Reasoning System) is a project aimed at the building of a general-purpose intelligent system, or a "thinking machine". This chapter focuses on the methodology of this project, and the overall development plan to achieve this extremely difficult goal. There are already many publications discussing various aspects of NARS, which are linked from the author's website. Especially, [1] is a recent comprehensive description of the project.

If there is anything that the field of Artificial Intelligence (AI) has learned in its fifty-year history, it is the complexity and difficulty of making computer systems to show the "intelligence" as displayed by the human mind. When facing such a complicated task, where should we start? Since the only known example that shows intelligence is the human mind, it is where everyone looks at for inspirations at the beginning of research. However, different people get quite different inspirations, and consequently, take different approaches to AI.

The basic idea behind NARS can be traced to the simple observation that "intelligence" is a capability possessed by human beings, but not by animals and traditional computer systems. Even if we accept the opinion that some animals and computers also show intelligence, they are still far inferior in this ability when compared to human beings.

If we can indeed draw the boundary of "intelligent system" to include normal humans, but not typical animals and computer systems, then the next question is: what is the difference between these two types of system? Hopefully the answer to this question can tell us what intelligence really is.

Even if this boundary of the category of intelligence is accepted, there are still many differences that can be found and justified. Now the problem becomes to identify a small number of them that are "essential", in the sense that they can derive or explain many other differences.

The design of NARS is based on the belief that the essence of intelligence is *the capability to adapt to the environment and to work with insufficient knowledge and* *resources* [2]. Therefore, an intelligent system should rely on constant processing capacity, work in real time, open to unexpected tasks, and learn from experience.

According to this opinion, whether a system is intelligent, or how intelligent it is, is not determined by what the system *can do*, but by what it *can learn to do*. An intelligent system does not need to be identical to the human brain in internal structure, or the human mind in external behaviors, nor does it have to provide optimal solutions to certain practical problems. Instead, the system should be general, flexible, and creative.

Though the above understanding of intelligence sounds simple, it does explain many phenomena observed in the human mind [1].

Similar to its conceptual basis, the engineering plan of NARS also follows "minimalism", in the sense that the goal is not to maximize the system's performance, but to minimize its theoretical assumption and technical foundation, while still being able to realize the conception of intelligence introduced above. The scientific reason for following this approach is that it produces a precise and verifiable theory; the practical reason is that it leads to a manageable project under the restriction of available development resources.

NARS is built in the form of a reasoning system, with a *language* for knowledge representation, a *semantic theory* of the language, a set of inference *rules*, a *memory* structure, and a *control* mechanism. The first three components are usually referred to as a *logic*. The reasoning framework has the advantage of being domain-independent, and combining the justifiability of individual inference steps and the flexibility of linking these steps together in a context-sensitive manner in run time.

In the following, the development plan of NARS is introduced first, which shows how the system is built stage by stage. After that, several important but optional augmentations of NARS are introduced and analyzed. Finally, several topics about the development of this system are discussed.

2. NARS Roadmap

The development of NARS takes an incremental approach consisting four major stages. At each stage, the logic is extended to give the system a more expressive language, a richer semantics, and a larger set of inference rules; the memory and control mechanism are then adjusted accordingly to support the new logic. Consequently, the system becomes more intelligent than at the previous stage, according to the previous conception of intelligence, by being more adaptive and more efficient in using available knowledge and resources.

The following description omits technical details and comparisons with other systems, which can be found in the other publications on NARS, such as [1].

2.1 Basic reasoning

At this stage, the system is equipped with a minimum *Non-Axiomatic Logic*.

The logic uses a *categorical* language called "Narsese", in which every statement consists of a *subject term* and a *predicate term*, linked by an *inheritance relation*. Intuitively, the statement says that the subject is a *specialization* of the predicate, and the predicate is a *generalization* of the subject. For example, "*bird* $\rightarrow$ *animal*" is such a statement, with "*bird*" as the subject term and "*animal*" as the predicate term. A "term", at the current stage, is just an atomic identifier. The inheritance relation is defined as from one term to another term, and as being reflexive and transitive in idealized situations.

For a given term, its *extension* is the set of its known specializations, its *intension* is the set of its known generalizations, and its *meaning* consists of its extension and intension. Therefore, given inheritance statement "*bird* → *animal*", "*bird*" is in the extension of "*animal*", and "*animal*" is in the intension of "*bird*".

It can be proved that a statement is true if and only if the extension of its subject is a subset of the extension of its predicate, and, equivalently, if the intension of its predicate is a subset of the intension of its subject.

Based on this result, statements can be made multi-valued to represent incomplete inheritance. For a given inheritance statement, its *positive evidence* is defined as the set of the terms that are either in both the extension of the subject and the extension of the predicate, or in both the intension of the predicate and the intension of the subject; its *negative evidence* is defined as the set of the terms that are either in the extension of the subject but not the extension of the predicate, or in the intension of the predicate but not the intension of the subject. Evidence is defined in this way, because as far as a piece of positive evidence is concerned, the statement is true; as far as a piece of negative evidence is concerned, the statement is false.

For a given statement, if the amounts of positive and total (i.e., positive plus negative) evidence are written as w^+ and w, respectively, then the truth-value of the statement is represented as a pair of real numbers in [0, 1], <*frequency*, *confidence*>, where $frequency = w^+ / w$, and $confidence = w / (w + 1)$. Consequently, truth-value in NARS uniformly represents several types of uncertainty, including *randomness*, *fuzziness*, and *ignorance* [1].

Defined in this way, Non-Axiomatic Logic uses an "experience-grounded" semantics, where the meaning of each term and the truth-value of each statement are determined according to the relevant experience of the system, by the system itself.

A statement with truth-value is called a "judgment". The inference rules of NARS are defined on judgments, and are designed and justified according to the above experience-grounded semantics. A typical inference rule is *syllogistic*, that is, it takes a pair of inheritance judgments as premises, which share exactly one common term, and the conclusion is an inheritance judgment between the other two terms. The position of the shared term determines the type of the inference, including the following (truth-values omitted):

Table 1.

	Deduction	**Abduction**	**Induction**
Premise 1	M → P	P → M	M → P
Premise 2	S → M	S → M	M → S
Conclusion	S → P	S → P	S → P

Each inference rule has an associated truth-value function, which calculates the truth-value of the conclusion from those of the premises, by determining the amount evidential support the conclusion gets directly from the evidence provided by the premises. An inference rule is *valid* if it can be justified according to the experience-grounded semantics. The details and justifications of the rules can be found in [1] and other publications on NARS.

There is a special rule used for *revision*, where the two premises contain the same statement, but are supported by evidence collected from different sources. The revision rule produces a conclusion, with the same statement as content, and a truth-value corresponding to the accumulated evidence from both sources.

A *belief* of the system is a judgment coming from, or derived according to, the system's experience. At this stage, a *task* that the system can carry out is either a question to be answered, or a piece of new knowledge (as a judgment) to be absorbed.

A belief can be used to answer questions, both of the "yes/no" type and the "what" type, as well as to derive new beliefs following the inference rules. For a given question, the syllogistic rules can also be used *backward* to produce derived questions, whose answers will lead to the answer of the original question, using *forward* inference.

Roughly speaking, the system's memory consists of a belief network, in which each node is named by a term, and each link corresponds either to a belief, or to a task. Each node with its incoming and outgoing links forms a *concept*.

When the system is running, usually there are many tasks under processing in parallel. The system assigns a *priority-value* to every concept, task, and belief. At each inference step, a concept is selected, and then a task and a belief are selected within the concept. All these selections are *probabilistic*, that is, an item with a higher priority-value has a higher probability to be selected.

With the selected task and belief as premises, applicable inference rules derive new tasks (and new beliefs, if the task is a judgment), and add them into the memory. When the memory is full, items (beliefs, tasks, and concepts) with the lowest priority values are removed to release their previous occupied space.

After each step, the priority-values of the involved concept, task, and belief are adjusted, according to the immediate feedback obtained in this step. In the long run, higher priority-values will be obtained by items that are more useful in the past history and more relevant in the current context.

For a given task, the system's processing course is a sequence of inference steps. Such a sequence is formed at the run time, does not depend on any predetermined task-specific algorithm, but is determined by many factors in the past experience and current context of the system. Since these factors change constantly, the system's processing path and response to a task is usually not a function of the task, and they are neither fully predictable in advance nor fully repeatable afterwards, given the task alone.

Designed in this way, the system is indeed adaptive and can work with insufficient knowledge and resources, though it can only handle simple tasks, because of its limited expressive and inferential power at this stage.

2.2 First-order reasoning

At the second stage, Non-Axiomatic Logic is extended by adding compound terms and variants of the inheritance relation into the language and the rules.

The *similarity relation*, "$\leftrightarrow$", is a variant of the inheritance relation ("$\rightarrow$"), and can be seen as symmetric inheritance. For a statement containing two terms linked by the similarity relation, the *positive evidence* includes the terms in the extension (or intension) of both terms, and the *negative evidence* includes the terms in the extension (or intension) of one term but not the other.

Inference rules involving similarity include *comparison* (such as to derive "S $\leftrightarrow$ P" from "S $\rightarrow$ M" and "P $\rightarrow$ M") and *analogy* (such as to derive "S $\rightarrow$ P" from "S $\leftrightarrow$ M" and "M $\rightarrow$ P"). Again, each rule has an associated truth-value function.

A *compound term* is formed by an *operator* from one or more *component terms*, and the composing process can be repeated to build compound terms with complicated internal structures.

The meaning of a compound term has a literal part and an empirical part. The former comes from the meaning of its components in a way determined by the operator; the latter comes from the role the compound term as a whole plays in the experience of the system, just like how the meaning of an atomic term is determined at the previous stage. In different compound terms, these two parts have different relative significances. The more the system knows about a compound term, the less it is used in its literal meaning.

The following compound terms are introduced into Narsese at this stage:

Sets. Though a term is not defined as a set in general, there are special compound terms corresponding to sets:

- An extensional set is defined by exhaustively listing its instances, such as in "$\{T_1, T_2, \ldots, T_n\}$".
- An intensional set is defined by exhaustively listing its properties, such as in "$[T_1, T_2, \ldots, T_n]$".

In both cases, "$T_1, T_2, \ldots, T_n$" are terms serving as the components of the compound, and "{}" and "[]" are operators for extensional set and intensional set, respectively.

Intersections and Differences. Compound terms can be formed via set operations on the extensions or intensions of existing terms. Using two different terms T_1 and T_2 as components, four compound terms can be defined in this way:

- "$(T_1 \cap T_2)$" is their extensional intersection. Its extension is the intersection of the extensions of T_1 and T_2, and its intension is the union the intensions of T_1 and T_2.
- "$(T_1 \cup T_2)$" is their intensional intersection. Its intension is the intersection of the intensions of T_1 and T_2, and its extension is the union the extensions of T_1 and T_2.
- "$(T_1 - T_2)$" is their extensional difference. Its extension is the difference of the extensions of T_1 and T_2, and its intension is the intension of T_1.
- "$(T_1 \varnothing T_2)$" is their intensional difference. Its intension is the difference of the intensions of T_1 and T_2, and its extension is the extension of T_1.

Products and Images. To represent ordinary relations among terms that cannot be treated as *inheritance* or *similarity*, compound terms can be used so that the statement is still about inheritance. For example, an arbitrary relation R among terms A, B, and C can be represented as an inheritance statement "$(\times\ A\ B\ C) \rightarrow R$", where the subject term is a *product* "$(\times\ A\ B\ C)$", which is a compound with three components in the given order, and the symbol "$\times$" is the *product operator*. To isolate each component out of the compound term, the above inheritance statement can be equivalently rewritten into any of the following form:

- "$A \rightarrow (\perp\ R\ \Diamond\ B\ C)$"
- "$B \rightarrow (\perp\ R\ A\ \Diamond\ C)$"
- "$C \rightarrow (\perp\ R\ A\ B\ \Diamond)$"

In each of the three inheritance statements, the predicate term is an *extensional image*, with "$\perp$" as the operator, and "$\Diamond$" as a placeholder indicating the location of the subject in the relation. Symmetrically, there is also a type of compound term called *intensional image*.

For each type of compound term defined above, there are corresponding inference rules for the *composition* and *decomposition* of compound terms. Consequently, the inference process not only produces new tasks and beliefs, but also generates new terms and concepts.

At this stage, the memory architecture and control mechanism of the system are revised to uniformly handle *compound* terms and *atomic* terms.

2.3 Higher-order reasoning

Higher-order reasoning allows statements to be used as terms. At this stage, the system can carry out reasoning on higher-order statements, i.e., statements of statements.

In the simplest situation, a higher-order statement is formed by an ordinary relation that takes a statement as argument. Such relations include "believe", "know", "say", etc. For example, in Narsese "Birds are animals" can be represented as a (first-order) statement "*bird* → *animal*", and "Peter knows that birds are animals" as a higher-order statement "(× {*Peter*} {*bird* → *animal*}) → *know*".

Another way to form higher-order statements is to use statement operators *conjunction* ("∧"), *disjunction* ("∨"), and *negation* ("¬"), as in prepositional logic, except that in Narsese the statements are multi-valued, rather than binary. Each of the three operators has an associated truth-value function when used to form compound statements.

Two new relations are introduced between two statements: *implication* ("⇒", intuitively meaning "if") and *equivalence* ("⇔", intuitively meaning "if-and-only-if"). In the implication statement "S ⇒ P", S is a *sufficient condition* of P, and P is a *necessary condition* of S. In the equivalence statement "S ⇔ P", S and P are *sufficient and necessary conditions* of each other.

Implication ("⇒") and *equivalence* ("⇔") are defined to be isomorphic to *inheritance* ("→") and *similarity* ("↔"), respectively. Consequently, some of the rules of higher-order inference are isomorphic to certain rules of first-order inference:

Table 2.

	Deduction	**Abduction**	**Induction**
Premise 1	M ⇒ P	P ⇒ M	M ⇒ P
Premise 2	S ⇒ M	S ⇒ M	M ⇒ S
Conclusion	S ⇒ P	S ⇒ P	S ⇒ P

Each of these rules uses the same truth-value function as its first-order counterpart.

There are also inference rules that are specially designed for higher-order inference, and have no direct counterpart in first-order inference.

The terms in Narsese defined so far are *constant*, in the sense that each term uniquely indicates a concept in the system. To extend the expressing power of the language, *variable terms* are introduced at this stage.

There are two types of variable: an *independent variable* is named by an identifier with a prefix "#", and indicates a unspecific term in the extension or intension of another term; a *dependent variable* is named by an identifier with a prefix "#" and followed by a list (which may be empty) of independent variables, and indicates a specific-but-unnamed term in the extension or intension of another term.

The scope of a variable term is one statement, so that the same variable term name in different statements may indicate different (constant) terms in the system.

A statement may contain multiple variables with embedded scopes. For example, the following sentences can be represented in Narsese using two variables, with difference relationships:

- "*Every key can open every lock"* is represented as:
  "(({#x} → key) ∧ ({#y} → lock)) ⇒ ((× {#x} {#y}) → open)"
- "*Every key can open a lock*" is represented as:
  "({#x} → key) ⇒ (({#y(#x)} → lock) ∧ ((× {#x} {#y(#x)}) → open))"
- "*There is a key that can open every lock*" is represented as:
  "({#x()} → key) ∧ (({#y} → lock) ⇒ ((× {#x()} {#y}) → open))"
- "*There is a key that can open a lock*" is represented as:
  "({#x()} → key) ∧ ({#y()} → lock) ∧ ((× {#x()} {#y()}) → open)"

There are inference rules responsible for the substitution between variable terms and constant terms in inference steps.

Again, the memory architecture and control mechanism are revised to uniformly handle *first-order* inference and *higher-order* inference.

At this stage, the expressive power of Narsese is comparable to the language of predicate calculus, in the sense that the two roughly overlap to a large extent, though neither is a subset of the other. The inferential power of the logic is extended to cover conditional inference, hypothetical inference, and abstract inference. Different from predicate calculus, all inference rules in NARS are designed according to the assumption of insufficient knowledge and resources, and justified according to the experience-grounded semantics described previously.

2.4 Procedural reasoning

Procedural reasoning means to infer about events, operations, and goals.

The first step is to introduce *time* into the logic. In the previous stages, the truth-value of a statement is timeless. At this stage, an *event* is defined as a special type of statement that has temporal truth-value, that is, its truth-value can change over time.

In Narsese, the temporal attribute of an event is represented *relatively*, with respect to another event. For two events, the simplest temporal relations are either they happen at the same time, or one of them happens before the other. The other temporal relations between them can be represented by dividing the involved events into sub-events, and further specifying the temporal relations among these sub-events.

By combining the two primary temporal relations with existing operators and relations, new operators and relations are formed, such as *sequential conjunction* (","), *parallel conjunction* (";"), *predictive implication* ("/⇒"), *retrospective implication* ("\⇒"), and *concurrent implication* ("|⇒"). For example, "S /⇒ P" indicates that S is a *sufficient precondition* of P, and P is a *necessary postcondition* of S.

With this representation, temporal inference can be carried out by separately processing the logical relation and the temporal relation in the premises, then combining the two results in the conclusion, such as in the following rules:

Table 3.

	Deduction	**Abduction**	**Induction**
Premise 1	M /⇒ P	P \⇒ M	M /⇒ P
Premise 2	S /⇒ M	S /⇒ M	M \⇒ S
Conclusion	S /⇒ P	S /⇒ P	S /⇒ P

O*peration* is a special type of event, which the system can actively *execute*, not just passively *observe*. In other words, an operation is a statement under *procedural interpretation*, as in logic programming. With the operators and relations introduced previously, the system can have beliefs on the preconditions (including causes) and postconditions (including consequences) of a given operation, and carry out explaining, predicting, planning, and skill learning, by reasoning on these beliefs.

The execution of an operation is usually accomplished by a mechanism outside the reasoning system, such as a hardware device with certain sensor/effecter capability, or another computer program with certain information-processing capability. The reasoning system interacts with them by issuing execution commands and collecting execution consequences, and these activities consist of the sensorimotor processes of the system as a whole. Operations toward the outside of the system correspond to *perception and control* of the environment; operations toward the inside of the system correspond to *self-perception and self-control*.

Operations are the means for the system to achieve *goals*. Before this stage, NARS can accomplish two types of tasks: knowledge to be absorbed, and questions to be answered. At this stage, a goal is defined as a statement to be made true (or as close to true as possible) by executing proper operations. A goal can be seen as an event, whose preconditions and postconditions will be gradually revealed through reasoning on available knowledge, and eventually related to operations.

The system uses a *decision-making* procedure to create new goals from desirable and achievable events. For this purpose, it attaches a *desirability-value* to each event, and includes their calculation as part of the inference process.

Furthermore, the memory architecture and control mechanism are revised to uniformly handle *goals/operations* and ordinary *tasks/beliefs*.

3. NARS and Beyond

The development of NARS has been carried out for more than two decades, roughly according to the plan described above. At the current time, the first three stages have been mostly finished, and the last stage should be completed in a few years. On-line demonstrations with working examples are available at the author's website.

After throughout testing and tuning as described in [1], NARS will be a general-purpose reasoning system, built according to the conception of intelligence as the ability to adapt with insufficient knowledge and resources.

However, that will not be the end of this research adventure. According to the above conception of intelligence, it is always possible to make a system more intelligent by extending its input/output channels to enrich its experience, or by improving its efficiency when using available knowledge and resources.

After NARS *per se* is fully built, there are still several major ways to augment the capability of the system. In the following, each of them will be briefly described, as well as analyzed to explain why it is not considered as a necessary part of NARS.

3.1 Sensors and effectors

For NARS, "sensors and effectors" are the input/output devises that allow the system to directly interact with the environment outside the language channel.

Such a device can be plugged into the system, according to the *operation* mechanism introduced in the previous section. Concretely, each usage of the device can be triggered by a command in the form of "(op, a_1, …, a_n)", where "op" is an operator (the action to be performed), and "a_1, …, a_n" are arguments of the operation (information needed in performing the action). To the reasoning system, this operation is inheritance statement "$(\times \{a_1\} \ldots \{a_n\}) \rightarrow op$" under procedural interpretation.

As explained before, the beliefs about an operation are represented mostly as its (pre and post) conditions. When the system concludes that a goal can be achieved by this operation, its degree of desirability is increased. If the operation is judged by the decision-making mechanism as sufficiently desirable and feasible, it becomes a goal itself, and it will be achieved directly by issuing the corresponding command, so that the device will perform the action to the (internal or external) environment. The reasoning system then will collect the feedback using its sensors to check if the goal has indeed been achieved, and to decide what to do next. A sensor is also invoked by corresponding operations, and it produces Narsese judgments, each of which states that an experienced item or pattern can be generalized by another one to a certain extent.

Because of the insufficiency of knowledge and resources, the system usually never knows all the preconditions and postconditions of each operation, and its expectations are not always confirmed by its future experience. Nor can it be guaranteed that the executed operations correspond to the optimal way to achieve its goals. Instead, the system just does its best, under the restriction of available knowledge and resources.

NARS is designed as a general-purpose system, without innate problem-specific tools, but these tools can be plugged into the system, and will be handled in a uniform manner. Consequently, NARS can be used either as an "intelligent operating system" with various software tools, or a "mind" of a robot with various hardware devices.

In artificial intelligence and cognitive sciences research, some authors stress the importance of sensorimotor to intelligence and cognition, and argue that the mind is situated and embodied [3, 4]. These opinions are agreeable, as far as they are taken to mean that thinking must be grounded in *experience*. However, it does not mean that thinking must be grounded in *human sensorimotor experience*. Since NARS is designed according to an experience-grounded semantics, it is situated and embodied. However, because the system is not designed to duplicate concrete human behaviors and capabilities, it is not equipped with human sensors and effecters. For example, there is no doubt that vision plays a central role in human cognition, and that computer vision has great practical value, but it does not mean that vision is needed in every intelligent system. Because of these considerations, sensors and effecters are treated as optional parts of NARS.

3.2 Natural languages

As mentioned previously, NARS uses Narsese, a formally defined language, to communicate with its environment. Since the language uses an experience-grounded semantics, the truth-value of each statement and the meaning of each term in the language are determined by the system's relevant experience. In this aspect, the language is very similar to natural languages.

On the other hand, since the grammar of Narsese is formally defined, it is very different from the grammar of any natural language. For NARS to use a natural language to communicate with human users, it needs to learn the grammar and lexicon of the language.

In NARS, natural-language learning can be carried out by the system's general-purpose learning mechanism on linguistic materials. Each word, phrase, and sentence of a natural language will be represented within the system by a term. These terms have a many-to-many mapping to the terms directly used in the system's "native language", Narsese, and this mapping corresponds to a *symbolize* relation in Narsese. The truth-value of a symbolizing statement indicates the frequency and confidence for the word/phrase/sentence (in the natural language) to be used as the *symbol* of the term (in Narsese), according to the experience of the system.

In language understanding process, NARS will not have separate parsing and semantic mapping phases, like in many other natural language processing systems. Instead, for an input sentence, the recognition of its syntactic structure and the recognition of its semantic structure will be carried out hand-in-hand. The process will start by checking whether the sentence can be understood as a whole, as the case of proverbs and idioms. If unsuccessful, the sentence will be divided recursively into phrases and words, whose sequential relations will be tentatively mapped into the structures of compound terms, with components corresponding to the individual phrases and words. If there are multiple candidates in the mapping process (i.e., ambiguity), the truth-values of the resulting statements will show which one is better supported, according to the system's experience.

The language production process is roughly the reverse of the understanding process. Driven by the goal of the communication activity, the ideas to be expressed will be selected or generated, then "translated" from Narsese into the target natural language, according to the learned symbolizing relation between the two languages.

In this way, NARS has the potential to learn and use any natural language, as far as its experience *in* that language can be related to its experience *outside* that language. However, due to the inevitable difference in experience, the system cannot always be able to use a natural language as a native speaker. Even so, its proficiency in that language should be sufficient for many practical purposes.

Being able to use any natural language is not a necessary condition for being intelligent. Since the aim of NARS is not to accurately duplicate human behaviors so as to pass the Turing Test [5], natural language processing is optional for the system.

3.3 Education

NARS processes tasks using available knowledge, though the system is not designed with a ready-made knowledge base as a necessary part. Instead, all the knowledge, in principle, should come from the system's experience. In other words, NARS as designed is like a baby that has great potential, but little instinct.

For the system to serve any practical purpose, extensive *education*, or *training*, is needed, which means to build a proper internal knowledge base (or call it belief network, long-term memory, etc.) by feeding the system with certain (initial) experience.

Various knowledge sources will be made available for NARS, and the possibilities include (though not limited to):

- Existing knowledge bases. NARS can be connected to existing knowledge bases, such as Cyc (for commonsense knowledge), WordNet (for linguistic knowledge), Mizar (for mathematical knowledge), and so on. For each of them, a special interface module should be able to approximately translate knowledge from its original format into Narsese.
- The Internet. It is possible for NARS to be equipped with additional modules, which use techniques like semantic web, information retrieval, and data mining, to directly acquire certain knowledge from the Internet, and put them into Narsese.
- Natural language interface. After NARS has learned a natural language (as discussed previously), it should be able to accept knowledge from various sources in that language.

Additionally, interactive tutoring will be necessary, which allows a human trainer to monitor the establishing of the knowledge base, to answer questions, to guide the system to form a proper goal structure and priority distributions among its concepts, tasks, and beliefs.

Unlike most of the training processes currently studied in machine learning, NARS cannot be trained to converge to certain predetermined stimulus-response mappings. Instead, the process will be more like the education of human beings, where learning is open-ended, and the tutors will have strong influence, but not complete control, of the system's behaviors. Therefore, the education theory for NARS will be similar to, though not necessarily identical to, that for human beings.

To improve the efficiency of education, it is possible to copy the memory of a trained NARS system into other NARS systems, so that they will not need to repeat the training process. Also, it is possible to directly edit the memory of a system to modify or implant certain knowledge. In theory, all of these shortcuts should be equivalent to certain possible experience of the system, so they do not conflict with the principle that all the knowledge of NARS comes, directly or indirectly, from experience.

One important issue to be handled through education is ethics. Unlike argued by some other researchers, NARS is not an attempt to design a “friendly AI”. As far as its initial state is concerned, the system is ethically neutral, since it can has any beliefs and goals. To make a NARS implementation “human friendly” means to give it certain beliefs and goals, which is an education mission, not a design mission. Even if something like Asimov’s “Three Laws of Robotics” is implanted into the system’s memory (which is possible), it still cannot fully control the system’s behaviors, due to the insufficiency of knowledge and resources in the system. What should and can be done is to educate an AI system like a child, by controlling its initial experience, to make it to love human beings, and to understand how to benefit the human beings, as far as its intelligence can tell.

NARS is designed according to the belief that the essence of “intelligence” is in the ability of flexibly acquiring, deriving, and applying knowledge, but not in possessing a large amount of human knowledge (as argued in [6]). For this reason, the building of a knowledge base for a specific practical application is an optional augmentation of the system.

3.4 Socialization

If multiple copies of NARS are implemented with different system parameters, hardware/software, and experience, they will form different beliefs and goals, and behave differently. However, as soon as they begin to communicate with each other, they will have common experience, and some consensus will be developed among the systems. Furthermore, these systems may begin to cooperate in the solving of certain complicated problems, as well as to compete for certain resources.

Since NARS uses the same language, Narsese, for input and output, the current design needs no major change to allow communication in a multi-system environment, which will lead to *socialization*, a continuing process whereby the system form and adjust its beliefs and goals as a consequence of interaction with other systems.

Like for human mind, socialization will play an important role in the mental development of an AI system, which is different from the role played by education, though both happen as consequences of communication. In education, the involved systems (the trainee and the trainer) have asymmetric status, while in socialization all the involved systems have similar status.

For an implemented NARS system, many concepts and skills can only be fully acquired through socialization, such as self/other distinction, speech action, game playing, and moral discipline.

Furthermore, with such a multi-system community as a model, many interesting topics can be studied. For example, since each system in the community is based on the same experience-grounded semantics, the dynamics of the community as a whole will show the forming and changing of common knowledge, which will provide insights about the evolution of language, science, and culture.

Even though socialization plays an important role in the growth of a mind (either natural or artificial), it is considered as optional for NARS. For a theoretical reason, it is because the notion of intelligence accepted in this project does not presume the existence of a society as part of the environment; for a practical reason, it is because to consider such a society is not required when the system is designed. On the other hand, unless each single system is properly designed, it will not be very fruitful to study how a multi-system community behaves.

3.5 Hardware

As a reasoning system, the running process of NARS consists of individual inference steps. Since the system is built around a *term logic*, it has the property that each inference step happens within a "concept", which collects beliefs and tasks about the same term. As a result, a concept can be naturally taken as a processing unit, which manages its own knowledge and resources. Different concepts independently execute a similar routine to carry out an inference step, and pass messages (tasks) to one another to cooperate in the processing of the tasks in the system.

Such a parallel-and-localized processing mode allows NARS to achieve higher performance and efficiency by running on specially designed hardware. Roughly speaking, such a hardware system will consist of a large number of processing units, each with its own processor and memory, corresponding to a concept in NARS. Each processing unit only needs to implement a simple algorithm, as required by an inference step, plus the algorithms needed for inter-concept communication and resources management. Such an implementation will surely achieve much higher time efficiency, compared to the current implementation in a general-purpose computer with a single CPU.

It is important to realize that even in such a special hardware implementation, the basic principle of NARS remains the same: the system still needs to work with insufficient knowledge and resources. Since the number of processing unit is a constant, and so does the capacity of each unit, they will need to be shared by the concepts, because the system as a whole will producing new concepts from time to time, whose number will soon exceed the number of processing units. Consequently, the system still need time-sharing and space-sharing, and it is only that what to be shared is not a single CPU and RAM, but many processing units.

Some people blame the von Neumann architecture of computer for the past failure of AI, but the argument is not convincing. It is true that the current computer architecture is not designed especially for AI, but it has not been proved that it cannot be used to implement a truly intelligent system. Special hardware is optional for NARS, since the system can be fully implemented on the current hardware/software platform, though special hardware will surely make it work better.

3.6 Evolution

Under the assumption of insufficient knowledge, all *object-level* knowledge in NARS can be modified by the system's various learning mechanisms. For example, the truth-value of any belief is revisable, and the priority-value of any belief, task, or concept is adjustable. All these changes are driven by the system's experience.

However, the current design of NARS does not support any automatic change of the *meta-level* knowledge of the system, which includes the grammar of Narsese, the factors determining truth-value and meaning, the inference rules, the resource-allocation mechanism, the priority-distribution strategy, as well as the values of system parameters. Meta-level knowledge is determined when the system is designed, and remains fixed during the system's life cycle, because any change at this level may destroy the system's consistency and integrity.

Obviously, a change in meta-level knowledge, that is, the system design, has the possibility of increasing the system's intelligence, though it is usually a dangerous experiment. For NARS, such changes are left for a separate *evolution process*, which will not be carried out by the reasoning/learning mechanisms in the current design.

The evolution of NARS will follow ideas similar to genetic algorithm [7]. First, a multi-dimensional design space will be determined, in which each dimension corresponds to a specific aspect of the system's design, and valid values on the dimension correspond to possible design choices for that aspect. In this way, a concrete system design can be represented by a point in the design space, whose coordinates form the "genetic code" of the system.

To search for the best design using genetic algorithm, a population of systems will be maintained, and in each generation, fitness values of the systems will be evaluated. When the next generation of system is produced, systems with higher fitness values will have a higher chance to be parents of new systems, which partially inherit the genetic codes of their parents. Some random mutation may also happen to the genetic code of a new system. In the long run, a natural selection process will produce systems with better fitness, which in this context means higher intelligence.

Though such an evolution process may indeed produce more intelligent NARS systems, it should not be confused with the experience-driven learning processes in NARS, which is what intelligence is about. In general, we should see *intelligence* and *evolution* as two different forms of *adaptation*. Roughly speaking, intelligence is achieved through experience-driven changes ("learning" or "conditioning") within a single system, while evolution is achieved through experience-independent changes ("crossover" or "mutation") across generations of systems. The "intelligent" changes are more justifiable, gradual, and reliable, while the "evolutionary" changes are more incidental, radical, and risky. Though the two processes do have some common properties, their basic principles and procedures are quite different.

4. Discussions

In this section, several issues about the development plan of NARS are discussed, in the context of the Artificial General Intelligence Research Institute Workshop.

4.1 Why to aim at a principle

The first major feature that distinguishes NARS from other AI projects is its conception of *intelligence* as the *principle* of *adaptation with insufficient knowledge and resources*. To most people, "intelligence" is more often associated with concrete behaviors, capabilities, and functions, than with general principles. This topic was initially addressed in [2], and has also been discussed in other publications on NARS, such as [1]. In the following, it is approached from a slightly different perspective.

As said at the beginning of the chapter, the boundary of "intelligent system" should be drawn in such a way, so as to include normal humans, but not typical animals and computer systems. On the other hand, we do not want to make the notion too narrow to take the human mind as the only possible form of intelligence. For example, to ask an AI system to be indistinguishable from human in behavior is too strong a request. Turing proposed his "Imitation Game" as a *sufficient* condition for a system to be intelligent, though he admitted that it might not be a *necessary* condition. He just thought that "if, nevertheless, a machine can be constructed to play the imitation game satisfactorily, we need not be troubled by this objection" [5]. Similarly, even though the human brain is indeed the only *known* way to produces intelligence, to take it as the only *possible* way to do that shows a lack of imagination.

On the other hand, to take intelligence as the ability of solving certain concrete problem seems to make it too easy, no matter how complicated the problem is. For example, the IBM computer system Deep Blue defeated the world chess champion, but many people still do not think it as intelligent, because it can do little beside chess. Indeed, almost all computer systems and animals can do something better than human beings, but they do not be called "intelligent" because of that. Otherwise, the concept of intelligence would be trivialized.

Many people enter the field of AI with a vague feeling that what we call "intelligence" in our everyday language has certain fundamental difference from how a conventional computer system works, and being more intelligent does not means to evaluate arithmetic expressions faster, to remember phone numbers for a longer period, or to accurately repeat a complicated procedures for more times. Instead, it is associated with features like generality, flexibility, and creativity. These features are displayed by an intelligent system in its problem-solving behaviors in various domains, but cannot, and should not, be bounded to a certain type of problem. Instead, they should be described as produced by a general *principle*, through a mechanism realizing the principle, which can be applied to various problems in various domains.

Such a "principle-based" conception of intelligence also helps for establishing the identity of a new research field, whether we choose to call it AI or AGI (Artificial General Intelligence). According to this definition, AI is different from theoretical Computer Science in that the latter mainly studies computer systems working with sufficient (though may be limited) knowledge and resources, while AI assume the opposite; AI is different from cognitive modeling in that the latter attempts to duplicate the details of human behavior, while AI only attempts to duplicate the general principle abstracted from concrete human behavior.

For practical reasons, almost all problems currently labeled as "AI" share the common nature that they need to be solved by adaptive systems working with insufficient knowledge and resources, and what we do not have yet is a normative theory for how to build such systems.

4.2 Why to take a unified approach

Intelligence, as displayed in the human cognition process, can be studied from various perspectives. Consequently, traditional AI research has evolved into many subfields, each has its problems and proposed solutions, such as search, reasoning, learning, planning, perception, action, etc. To build an AGI, it is very natural to adopt an *integrative approach*, by combining the most premising solution in each subfield to cover all aspects of intelligence.

Another justification of the integrative approach is that since every AI techniques has its strength and weakness, to use all of them together should give us the best solution. Consequently, people have attempted to integrate multiple techniques, like rule-based, case-based, statistical, connectionist, evolutionary, robotic, etc., into a hybrid system, in which each technique is used for what it is best for.

The above opinions are all reasonable, and research projects on integrative AGI have produced interesting results, as described in several chapters of this volume. However, it is not the only possibility.

NARS does not take an *integrative* approach, but a *unified* approach. Though its design has been influenced by the ideas of various schools in AI and Cognitive Sciences, the resulting technique is a singleton, rather than a "toolbox" consists of multiple independently developed tools.

It is possible for NARS to follow a unified approach, first because of its aim. As discussed previously, in this research "intelligence" is seen as a principle, not a group of loosely related capabilities or functions. It is much more likely to use a single technique to realize a principle, than to realize multiple independent capabilities.

Furthermore, the unified approach is adopted in NARS, not because it is easier than the integrative approach, but because it has theoretical and practical advantages.

Scientific research always prefers unified explanation of complicated phenomena. Driven by similar motivations, many previous AI projects attempted to use a single technique to cover most of the field, if not all of it. General Problem Solver tried to treat all problem-solving processes as heuristic search in a state space [8]; the Fifth

Generation Computer System was based on the assumption that many AI problems can be solved by parallel inference [9]. Their failures make many people to believe that AGI cannot be achieved by a single technique, given its well-known complexity. However, this kind of evidence is far from conclusive. The previous failures may be caused by their selections of aims and/or techniques, and they cannot rule out the possibility that when the aim of the research is properly set, certain technique can achieve it. Also, a system based on a single technique can still produce very complicated behavior, given the complexity of the environment of the system.

From an engineering point of view, the most difficult part of the integrative approach is not in the building of individual "modules" using different techniques, but in organizing them into a whole [10]. To passing data from module to module is easy, but it is hard to interpret the data in a wide range of situations consistently in different modules, because the involved techniques have been developed on very different theoretical foundations. When an integrative/hybrid system does not work properly, it is not easy to say what is wrong, because the lack of a common basis of the system. On the contrary, the development and evaluation of a unified system is much better guided by the theory behind the technique. Even if it eventually fails to cover the whole field, we can still get a much more clear picture about the capability and limitation of the technique, compared to the situation where the technique is just one of several in the system.

Since NARS can use various techniques as tools for practical problems, it will not attempt to directly use a single technique on all kinds of problems. Instead, the heterogeneous tools will be handled in a unified manner. For a concrete problem, the system will prefer to apply an available tool, and it is when no such a tool is there or can be easily made, will the system try to solve it using the reasoning ability.

Finally, even though human intelligence can be studied from different perspective, there is still evidence suggesting the close relationship of the various facets. As Piaget commented, intelligence "appears as a total system of which one cannot conceive one part without bring in all of it" [11].

In summary, now it is still too early to decide whether AGI is better pursued by an integrative approach or a unified approach. What is argued above is that the unified approach is not as obviously wrong as many people currently believe.

4.3 Why to work in a reasoning system

Even among people who agree that AGI may be achieved using a single technique, reasoning system, or logic-based system, is still not favorable, for various reasons [12]. Typical objections include:

- "Reasoning must be applied on materials provided by a sensorimotor mechanism, which was also evolved before high-level cognition."
- "Logic is too abstract, while real intelligence should be grounded, situated, and embodied."
- "Logic is too rigid to capture the flexibility of intelligent behaviors, which are often irrational."
- "Reasoning technique has been studied in AI for decades, and the results are not promising."

These opinions are all acceptable, as far as the involved "logic" refers to first-order predicate logic or its variants, such as non-monotonic logics. However, people often forget that in its original and broad sense, "logic" is just the attempt of capturing valid patterns of inference in a content-independent manner, and "inference" is just the process by while new knowledge is derived from existing knowledge. First-order predicate logic was originally designed to provide a logic foundation *for mathematics*, so if it cannot be used to capture the patterns of human thinking in general, it still does not mean that no logic can do that.

The logic implemented in NARS, Non-Axiomatic Logic, is fundamentally different from traditional mathematical logic, in that it is an attempt to capture the principle of adaptation with insufficient knowledge and resources. In this logic, a "term" is an identifiable item or pattern in the system's experience; a "statement" is an relation between two terms indicating their substitutability; the "truth-value" of a statement measures how the statement is supported or refuted by the system's experience; the "meaning" of a term indicates the role it plays in the system's experience; the function of an "inference rule" is to accomplish a single inference step, which build term(s) and/or statement(s) to summarize the information in existing ones; and a "reasoning process" is a sequence of step carrying out the tasks in the system for surviving and adapting. Since these notions are used in such broad senses, in NARS "reasoning" covers many activities that are usually called by other names, such as "learning", "planning", "perceiving", and so on.

Such a system is situated and embodied, because truth and meaning are grounded in the system's experience, and sensitive to the change of context. The behaviors of the system are "rational" with respect to its available knowledge and resources, though there is no guarantee that all of its predictions will be confirmed by its future experience, or to be judged by an observer as "rational" according to knowledge and resources that are unavailable to the system.

The reasoning technique used in NARS not only solves or avoids many problems in traditional logic-based AI systems [1], but also has many advantages over competing techniques (such as problem-specific algorithms, Bayesian networks, connectionist models, genetic algorithms, and reactive robots):

- It uses a domain-independent language for knowledge representation, which is more expressive than problem-specific data structures or numerical vectors in coding items and patterns in the system's experience.
- It uses an experience-grounded semantics to associate the terms and statements to the items and patterns in the (symbolic and/or sensorimotor) experience of the system.
- It uses inference rules to regulate individual steps of its working process, and each rule is justifiable in a domain-independent manner, according to the same semantics.
- It allows the system to organize the individual steps into task-processing courses at run time, so as to achieve flexibility and creativity. Consequently, the system can handle novel tasks by the cooperation of reliable basic actions, according to experience and context.

Though building a reasoning system is not necessarily the only way to AGI, there are reasons to say that it seems more promising than the other proposed alternatives.

4.4 Why to have these stages and augmentations

As described earlier, the development if NARS can be roughly divided into four stages, each of which is built on the top of the previous ones. After completion, NARS can be further augmented in (at least) six different directions, which are largely independent of each other.

This plan makes the development of the system *incremental*, in the sense that in each phase of the project, the immediate aim is relatively simple and clear, so it is possible to be carried out with a small expense. After each phase, the system is “more intelligent”, in terms of its ability of adapting to its environment and its efficiency of using its knowledge and resources.

Though this incremental development approach builds the system one part as a time, it is not “integrative”, since each stage is not built in isolation, but on top of the previous parts. Each of the six augmentations is indeed developed independent of each other, but is still based on NARS as a whole.

On the other hand, this dependency is one-way, since an earlier stage in NARS does not depend on any later stage, and the design of NARS as a whole does not depend on any of the augmentations. As discussed previously, though each augmentation makes the system more intelligent, and has important practical application, it is optional for an AGI system. On the contrary, NARS is the common core of any AGI system developed by following this research plan.

Almost for every augmentation listed before, there are people who believe either that it is more crucial than the ability of reasoning for an AGI, or that it should be taken as a necessary part of any AGI, together with reasoning. If we analyze each of these beliefs in detail, we will see that it is usually based on a conception of intelligence that identifies it with certain human behavior or capability. As stated before, such a conception tends to define intelligence too close to its only known example, human intelligence, to explore the possible forms of intelligence.

In that case, why the ability of reasoning is absolutely necessary for an AGI? As discussed above, in NARS the notion of “reasoning” is extended to represent a system’s ability to predict the future according to the past, and to satisfy the unlimited resources demands using the limited resources supply, by flexibly combining justifiable micro steps into macro behaviors in a domain-independent manner. There are reasons to believe that any AGI needs this ability.

Because of this, NARS is not merely a reasoning system, but an attempt to model the core of any thinking machine in an abstract manner. To satisfy additional requirements, we may also plug various sensors and effecters into the machine, teach it natural languages, train it with commonsense and domain-specific knowledge, raise it in a multi-system community, run it in specially-designed hardware, or let it evolve from generation to generation. Even so, we still need to build the core first.

Acknowledgement

The author thanks the anonymous reviewer for helpful comments.

Reference

[1] P. Wang, Rigid Flexibility: The Logic of Intelligence, Springer, 2006.
[2] P. Wang, On the Working Definition of Intelligence, Technical Report No. 94, Center for Research on Concepts and Cognition, Indiana University, 1994.
[3] L. Barsalou, Perceptual symbol systems, Behavioral and Brain Sciences 22 (1999), 577-609.
[4] R. Brooks, Intelligence without representation, Artificial Intelligence 47 (1991), 139-159.
[5] A. M. Turing, Computing machinery and intelligence, Mind LIX (1950), 433-460.
[6] D. Lenat and E. Feigenbaum, On the thresholds of knowledge, Artificial Intelligence 47 (1991), 185-250.
[7] J. Holland, Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence, MIT Press, 1992.
[8] A. Newell and H. A. Simon, GPS, a program that simulates human thought, E. A. Feigenbaum and J. Feldman (editors), Computers and Thought, 279-293, McGraw-Hill, 1963.
[9] E. Feigenbaum and P. McCorduck, The Fifth Generation: Artificial Intelligence and Japan's Computer Challenge to the World, Addison-Wesley, 1983.
[10] A. Roland and P. Shiman, Strategic Computing: DARPA and the Quest for Machine Intelligence, 1983-1993, MIT Press, 2002.
[11] J. Piaget, The Origins of Intelligence in Children, W.W. Norton, 1963.
[12] L. Birnbaum, Rigor mortis: a response to Nilsson's "Logic and artificial intelligence", Artificial Intelligence 47 (1991), 57-77.

Advances in Artificial General Intelligence: Concepts, Architectures and Algorithms
B. Goertzel and P. Wang (Eds.)
IOS Press, 2007
© 2007 The authors and IOS Press. All rights reserved.

Adaptive Algorithmic Hybrids for Human-Level Artificial Intelligence

Nicholas L. CASSIMATIS
Department of Cognitive Science, Rensselaer Polytechnic Institute

Abstract. The goal of this chapter is to outline the attention machine computational framework designed to make a significant advance towards creating systems with human-level intelligence (HLI). This work is based on the hypotheses that: 1. most characteristics of human-level intelligence are exhibited by some existing algorithm, but that no single algorithm exhibits all of the characteristics and that 2. creating a system that does exhibit HLI requires adaptive hybrids of these algorithms. Attention machines enable algorithms to be executed as sequences of attention fixations that are executed using the same set of common functions and thus can integrate algorithms from many different subfields of artificial intelligence. These hybrids enable the strengths of each algorithm to compensate for the weaknesses of others so that the total system exhibits more intelligence than had previously been possible.

Keywords. Human-level intelligence, Cognitive architectures

1. Motivation

The goal of this chapter is to outline a computational framework that makes a significant, measurable advance towards creating systems with human-level intelligence (HLI). This work is based on the hypotheses that: 1. most characteristics of human-level intelligence are exhibited by some existing algorithm, but that no single algorithm exhibits all of the characteristics and that 2. creating a system that does exhibit HLI requires *adaptive hybrids* of these algorithms.

1.1. Why Human-Level Intelligence

In this chapter, a system will be said to have *human-level intelligence* if it can solve the same kinds of problems and make the same kinds of inferences that humans can, even though it might not use mechanisms similar to those humans in the human brain. The modifier "human-level" is intended to differentiate such systems from artificial intelligence systems that excel in some relatively narrow realm, but do not exhibit the wide-ranging cognitive abilities that humans do. Although the goal is far off and there is no formal characterization of human cognitive ability, there are several reasons for adopting it:

1. Assuming humans are entirely composed of matter that can be simulated by computers, then if a human can do X, a (sufficiently powerful but still physically realizable) computer can do X also. Thus, while factoring million-bit numbers in a few seconds may be beyond the reach of any physically realiz-

able computer, activities such as having a human-language conversation or discovering a cure for a disease are not.
2. The applications of human-level intelligence would be tremendous.
3. Ideas behind a human-level artificial intelligence are likely to help cognitive scientists attempting to model human intelligence.
4. Decisions in designing intelligent systems often require tradeoffs, for example between soundness and completeness vs. quick computation. Since human beings are an existence proof that a system can be quite powerful without sound and complete planning and decision making, we know that a system that does not offer such guarantees would nevertheless be quite powerful. Thus, the human case can provide motivation or justification for making certain tradeoffs.

1.2. Problem of Tradeoffs

There are several characteristics of AI systems we want. The problem is that in the case of human-level intelligence, most existing algorithms exhibit some characteristics at the expense of others. For example:

Generality vs. speed. Many search, Bayes network and logic-theorem proving algorithms are often (in the limit, at least) guaranteed to produce a correct answer. This makes them very general. However, for problems involving many state variables, these algorithms are often too slow. More "structured" algorithms such as case-based reasoning or script matching can produce quick results even on problems with enormous state spaces. However, these algorithms often have trouble when there is no good case or script that matches a new situation. They thus trade speed for generality.

Complex vs. robust behavior. "Traditional" planning algorithms are capable of constructing complex sequences of actions to achieve goals. However, in a system with noisy sensors in a changing world, plans are often invalidated in the time it takes to formulate them. Reactive systems (e.g., [1,2]) quickly react to an existing situation while often not creating or manipulating any model of the world. This, however, is a problem in situations that require complex plans about unseen, past and/or future parts of the world. Reactive systems therefore often trade flexibility for complexity.

These sorts of tradeoffs are not an obstacle in many domains. For example, problems that can be formulated with a modest and fixed number of state variables can often be successfully and quickly solved using SAT-based search algorithms. In this case, speed and generality are both possible.

However, many problems that humans can solve are so large and open-ended that tradeoffs in existing algorithms seem to become difficult to avoid. For example, a system that must read, summarize, make inferences from and answer questions about press reports on, for example, the biotech industry, faces several problems:[1]

Large state space/knowledge base. The amount of background knowledge that can be brought to bear in any particular story is enormous. Understanding and making inferences from a sentence such as "The discovery of the botulism vial in Sudan just before Thanksgiving has increased the value of companies targeting proteases enzymes" requires knowledge of biology, terrorism, drug discovery, stock prices and US Holi-

[1] The progress of text retrieval, summarization and question-answering systems in this domain do not mean this is a solved problem. These systems are still at best approximate human experts and are not nearly as capable as they are.

days. Any system that makes human-level inferences based on such text must therefore operate in a state space with an enormous number of state variables. This is often true for many problems humans can solve. Such large state spaces make it very difficult or impossible to avoid tradeoffs between generality and speed.

Time and equality. State variables can change over time. There are on the order of a billion seconds in a human lifetime. This significantly exacerbates the problem of state-space size. A similar problem is caused by identity. We often cannot perceive, but must infer, that an object seen now is the same as an object that was seen earlier. The possibility of such identities can greatly increase a state space.

Open world. Even in the simplest scenarios, inferring the existence of previously unknown objects is routine. For example, someone seeing a ball on a table roll behind an occluding object and not emerge from behind it can infer that there must be something like a hole, obstacle or adhesive substance which halted the ball's motion. However, many algorithms assume that the objects that exist are specified and fixed before reasoning begins. This is the case for many search, Bayesian reasoning and logic-theorem proving algorithms. When this assumption does not hold, these algorithms often either makes them inefficient or inoperable. For example, it is difficult to stop searching for a proof if adding another object to the universe is an option an algorithm has before giving up.

Changing world and noisy sensors. New sensor readings or changes in the world often invalidate an algorithm's inferences or choices. One option is for algorithms to be rerun each time information changes. This places a severe time constraint on them since they must be ready to rerun as soon as new information comes in. Another option is for algorithms to be modified to take new information into account as they are being executed. This is a much harder problem given how most complex algorithms are implemented today.

1.3. Uniform and Modular Approaches to the Tradeoff Problem

There are two common ways to avoid making these tradeoffs. First, extend an existing computational method so that it does not make a trade. For example, dynamic Bayes networks the fact that state variable values can change over time. DBNs thus reduce the expressive power one must trade to achieve (approximately) probabilistically correct inference. There are many similar efforts in many subfields, but the work in this project is based on the hypothesis that the drawbacks of individual classes of algorithms cannot ever completely be removed and that at a minimum it is worth exploring other alternatives to algorithmic uniformity.

One such alternative is to create computational frameworks for combining modules based on different data structures and algorithms. One potential problem with these approaches is that the tightness of integration between algorithms and data structures may not be strong enough. For example, it is conceivable that grounding a literal in a logic theorem proving module might be accomplished by using a vision system, a neural network and/or database access. While the work in this project employs some modular integration, it is based on the hypothesis that modular integration alone will not be sufficient and that, somehow, every single step of many algorithms can and must be integrated with many other algorithms in order to achieve HLI.

Both approaches – extending a single computational method and modular architectures – have achieved significant success. It is difficult to decisively argue that the problems just mentioned cannot be overcome. Nevertheless, since these approaches are

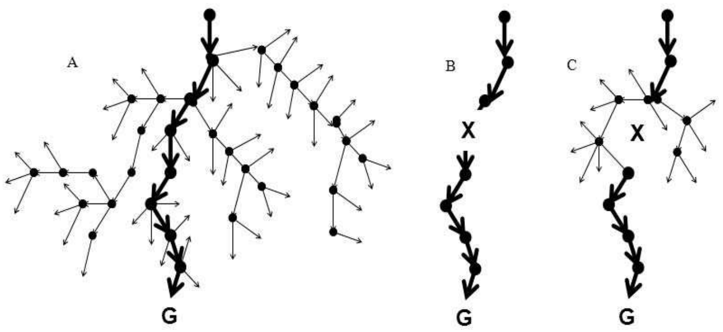


Figure 1. A hybrid (C) of systematic, general, flexible but slow search (A) with fast but rigid case-based reasoning (B).

already being studied by so many, this chapter explores the possibility of another approach to resolving tradeoffs between different computational methods.

1.4. Adaptive Hybrids

The overall approach of this work is not only to create systems that involve many algorithms in modules, but to execute algorithms that exhibit the characteristics of multiple algorithms depending on the situation. An example of what this might look like is illustrated in Fig. 1. Figure 1a depicts a search tree that finds a plan. The "bushiness" of the tree illustrates how search is slowed down by having to try many possible options. Figure 1b illustrates a case-based planner retrieving this plan when a similar situation arises. The "X" illustrates an imperfection in a plan (i.e., an obstacle to its successful application) and illustrates the potential rigidity of a pure case-retrieval approach. Figure 1c illustrates a hybrid of search and case-based reasoning that correctly solves the problem. Most of the problem is solved by case-application, but in the region where there is a problem with the case, search is used to solve the problem.

The goal is to create systems that execute such hybrids of algorithms and adapt the "mix" of the algorithms to changing characteristics of the situation. The intended result of this is the ability to create systems that exhibit more characteristics of HLI together in one system than has been possible so far. In addition to demonstrating the power of the approach on challenge problems in microworlds, we also intend to construct systems that measurably produce a significant advance in at least one important application domain.

2. Algorithms as Different Ways of Exploring a Multiverse

In order to motivate the design of a computational system that implements hybrid algorithms, it is helpful to conceive of these algorithms within a common formal framework. Developing such a framework would be key component of the work in this project. This section provides a preliminary sketch which motivates the *attention machines* presented in subsequent sections. Although much of the terminology here is borrowed from first-order logic and probability theory, this project is not an attempt to reduce all of AI to one or both. These are merely formalisms for *describing* the goals and capabilities of agents and algorithms. Attention machines are intentionally designed to include methods not normally associated with logic or probability theory.

2.1. Multiverse

Intuitively, a *multiverse* is the set of all possible worlds. Each world includes a history of past, current and future states. Multiverses are specified in terms of propositions. $R(a_1, \ldots, a_n, w)$ states that relation R holds over the a_i in a possible world, w. Arguments and worlds are drawn from the set of *entities, E*, where W, a subset of E, is the set of possible worlds in the multiverse. A multiverse is a triple, (M, E, W), where M is the set of all true propositions. $R(a_1, \ldots, a_n)$ is said to be true in world, w, if $R(a_1, \ldots, a_n, w)$ is a subset of M. The set of worlds is a subset of the set of entities so that one can state declarative facts about worlds, for example, to say that one world is counterfactual relative to another. Worlds are related to, but different from situations in the situation calculus. Situations conflate a representation of time and possibility. There is no temporal order among worlds. Each world "contains" a full (version of a) history.

2.1.1. Regularities in a Multiverse

Regularities that knowledge representation schemes capture can be reflected in a multiverse. For example, the logical formula, $(x)\ (R(x) \rightarrow S(x))$ can be modeled by a multiverse in which, for every world w, where $R(e)$ is true for some e, $S(e)$ is also true. Probabilistic regularities can be modeled by treating all the possible worlds as equiprobable. Thus, $P(R(x)) = |W_r|/|W|$, where W_r is the set of all worlds where $R(x)$ is true. Conditional probabilities can be similarly modeled.[2]

2.1.2. Formulating the Goals of Algorithms in the Multiverse

It is possible to characterize the goals of algorithms from different computational frameworks using multiverses. This will enable a characterization of their operation that will motivate a computational approach for executing hybrids of these algorithms. This chapter will choose simple versions of problems from multiple subfields of artificial intelligence to illustrate this point.

Search. There are so many search algorithms that it is difficult to give a single characterization of what they are all designed to do. Many search algorithms can be thought of as methods for finding a state of affairs that satisfy some constraint. This is the purpose of most SAT-based search algorithms (see ([3]) for a review) and many other search problems can reduce to searches for models that satisfy constraints (e.g., STRIPS planning ([4])). In terms of the multiverse, these search algorithms try to find a possible world where these constraints are satisfied. In many typical single-agent planning contexts, for example, search aims to find a possible world in which a goal constraint is satisfied at a time in the future such that each difference between that future state and the current state results from an action of the agent, directly or indirectly.

Graphical models. Graphical models are used to capture conditional probabilities among state variables and to compute the probability distribution over a set of variables given observed values of those variables. Bayesian networks ([5]) are perhaps the best-

[2] This treatment of probabilistic relationships, clearly applies only when the set of all possible worlds is finite, though extending this approach to multiverses with infinitely many possible worlds should be straightforward.

known class of graphical models. A node in a network representing state variable *X* set to value *a* can be represented with the proposition *Value(X,a)*. Of course, there is no reason that a more expressive scheme cannot be adopted for specific applications (e.g., using *Color(dog,black)* to represent that the state variable corresponding to a dog's color state variable being set to 'black'). The prior and conditional probabilities in a Bayes Network (which correspond to edges in the network) can be captured by treating possible worlds in a multiverse as equiprobable, as outlined in the previous subsection. In multiverse terminology, the aim of a Bayes Network propagation algorithm is to find, for each state variable *X* and value *v*, the proportion of possible worlds where $X = V$. Thus, when one of these algorithms determines that $P(X = a) = p$, it is estimating or computing that *Value(X,a)* is true in $p*N$ possible worlds, where N is the total number of possible worlds.

Logic programming. Logic programming systems enable knowledge to be declared in a logical formalism and queries about what that knowledge entails to be automatically answered, often by searching for proof of a theorem. The clauses and literals which make up logic programs are straightforwardly mapped onto a multiverse. The last subsection showed how a multiverse could capture material implication (clauses).[3] Thus, the goal of a logic programming algorithm determining whether a closed literal *L* is true is to somehow (e.g., by searching for a proof or failing to find a model where *L* is false) show that that proposition P_L representing *L* is true in all possible worlds in the multiverse.

2.2. The Execution of AI Algorithms as Paths Through the Multiverse

It is possible to characterize, not only the goals, but the operation of algorithms from multiple subfields of AI within the same multiverse. This motivates a computational architecture for implementing hybrids of these algorithms. The key idea, which we call the **common operation principle**, is to recognize that these algorithms can be decomposed into the same set of "common operations" which each involve exploring some part of the multiverse. A provisional set of common operations developed in preliminary work include:

Forward inference. Given the truth values of a set of propositions, *P,* produce a set of propositions, *Q*, entailed or made more likely by the those truth values.

Subgoaling. Given the goal of determining whether a proposition *Q* is true, produce a set of propositions, *P*, whose truth values would make *Q* more or less likely.

Grounding. Given a proposition, *P*, with open variables, return a set of true closed propositions that are identical to *P* under some assignment.

Identity and similarity matching. Given an entity, *e*, and a set of true and false propositions about *e*, return a set of entities *E* that are similar or (likely to be) identical to *e*.

Exploring possible worlds. Given a world, *w*, and a set of proposition-truth value pairs, return a world w_1 such that those propositions have those truth values and where otherwise everything that is true or false in *w* is so in w_1.

[3] Literals with functions, e.g., which Prolog permits and Datalog does not, can be flattened to correspond to literals in a multiverse.

When an algorithm performs one of these operations on a proposition, we say that it *attends to* or *fixes its attention* on this proposition. We will also call this event an *attention fixation.* A key insight motivating this chapter is that **AI algorithms from different subfields based on different computational formalisms can all be conceived of as strategies guiding attention through propositions in the multiverse.**

Search. We illustrate how to frame search in the multiverse context using GSAT ([6]) because it is relatively simple, though extending this approach to other algorithms is straightforward.

Goal: Find a possible world where constraint *C* is true.
For MAX-TRIES:

- Start with a set of propositions, *P*, which can be true or false.
- For MAX-FLIPS
- Choose a world, *w*, by choosing a random subset of *P* to assign as true.
- If *C* is satisfied in *w*, then return *w*. *Forward inference.*
- Choose the proposition in *P* whose truth value changing will lead to the greatest decrease in the number of clauses in *C* being unsatisfied. *Subgoaling.*

In other words, GSAT, like search algorithms generally, explores the possible worlds in a multiverse. At every step, it chooses which world to explore by applying *forward inference* to the currently explored world to see if it is true (i.e., given truth values for propositions in a world, infer whether *C* holds in that world) and subgoaling (i.e., given the goal of making *C* true, find the propositions in the current world that will bring C closer to being true). Thus, GSAT can be characterized as a sequence of possible world exploration, forward inference and subgoaling in the multiverse.

Rejection sampling. Rejection sampling, an algorithm for probabilistic inference, can be straightforwardly recast as a method for exploring worlds in a multiverse in the following manner (recall that every node, *X*, in a network corresponds to an open proposition *Value(X,?value)*):

For *N* worlds, $w_1 \ldots w_n$:

- Every edge node, *E*, with an unknown value corresponding to proposition *Value(E,?value,*w_i*)* grounds its value by sampling from the prior probability distribution corresponding to that edge. *Grounding.*
- Repeat until every state variable in the network has been set:
- For each interior node, *I*, whose input variables have been set:
- Given the input nodes and the conditional probability distribution associated with the edges leading into a node, sample a value, *V*, for *I*, yielding the proposition, *Value(I,V,*w_i*)*. *Forward Inference.*
- If the sampled value conflicts with an observed value stop exploring this world.

Estimate $P(X = v) \approx |\{w : Value(X,v,w)\}|/N$, where *N* is the number of possible worlds not thrown out during stochastic simulation.

Logic theorem proving. Many logic programming systems are based on algorithms such as SLD-resolution that repeatedly set subgoals and ground open literals. These algorithms can be straightforwardly formulated in terms of the common operations described above ([7]).

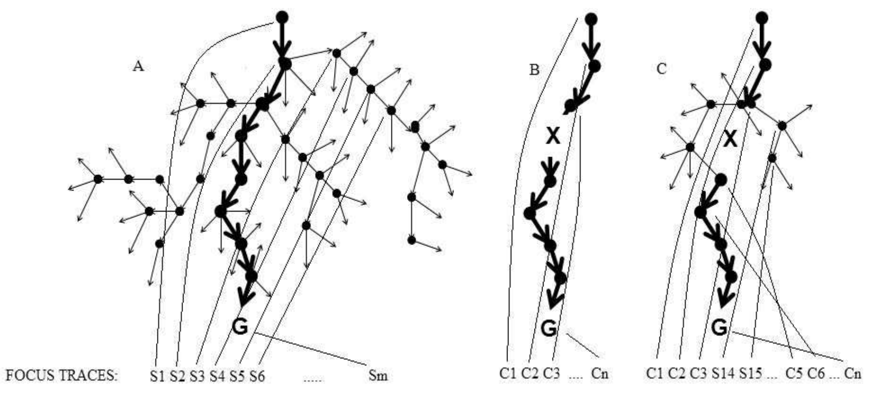


Figure 2. Algorithms and their focus traces. The focus trace (C) for the hybrid of case-based reasoning and search is a combination of the focus traces for search (A) and case-based reasoning (B).

- *Case-based reasoning.* Consider the use of case-based reasoning to find a plan for dealing with a novel situation.
- Goal: Find an action, *a*, that will bring about effect, *e*, i.e., *Cause(a,e)*.
- Find an event in the past, *e'*, and action *a'*, in the past such that a' caused e' (*cause(a',e')*), *e'* is similar to the currently desired effect, *e*, (*Similar(e,e')*) and such that *Similar(e,e'). Identity match.*
- In the world, w_m, where *Cause(a,e)* and *Similar(a,a')* (i.e., the world where the action taken to cause *e* is similar to the remembered action, *a'*). *Explore possible world.*
- For each of the roles or slots (r_1' ... r_n') of *a*, find entities (r_1... r_n) such that *Similar*(r_i', r_i,w_m). *Identity match.* (For example, if *a'* is the action of pounding a nail with a hammer, two roles are the hammer, *a*, and the nail, *n*. If the current goal is to insert a tent state into the ground, and the case-based reasoner decides to drive the stake into the ground with a rock, this is expressed by *Similar(hammer,rock,*w_m*)* and *Similar(nail,stake,*w_m*)*.)

Thus, a simple case-based reasoner can be constructed out of a combination of the similarity matching and possible world exploration common operations.

2.3. Hybrid Algorithms Through Hybrid Focus Traces

Each of these algorithms attends to (i.e. performs a common operation on) one proposition at a time. The sequence of attention fixations an algorithm makes when it executes is called its *focus trace*. Focus traces thus provide a uniform way of characterizing the execution of algorithms from different computational methods. This is a direct result of the common operation principle.

Focus traces also motivate an approach to executing hybrids of algorithms. One way to think about a hybrid of two algorithms is to think of hybrids of their focus traces. That is, an algorithm, H is a hybrid of algorithms A1 and A2 if H's focus trace includes fixations from both A1 and A2. For example, Fig. 2 illustrates the hybrid execution of case-based reasoning and search. The focus trace (2c) for the hybrid of case-based reasoning and search is a combination of the focus traces for search (2a) and case-based reasoning (2b).

Thus, if we can create a computational architecture that selects common operations from multiple algorithms, it would lay down a focus trace that would be a hybrid of those algorithms' focus traces. The next section sketches the *Attention Machine* architecture for executing hybrids in this manner.

3. Attention Machines

Attention machines are systems that execute algorithms by sequences of common operations. Each common operation is implemented as an attention fixation. Attention machines enable hybrids of algorithms to be executed by interleaving sequences of attention fixations.

Formally, an attention machine (AM) is an ordered pair (S, FM), where S is a set of modules called *specialists* and FM is a *focus manager*.

The reason for including multiple specialists in an AM is that each common operation can be implemented using multiple data structures and algorithms. We call this the *multiple implementation principle.* This principle enables modules to compensate for each other's weaknesses. For example, neural networks are often better at making forward inferences about object categories than rule matchers, while rule matchers are often better at making forward inferences involving causal change than neural networks. Thus, depending on the situation a particular common operation may be better performed using one computational method over another. Having several computational methods (each encapsulated inside specialists) implement each common operation increases the chances that a common operation will be successfully performed. What happens when two specialists conflict in performing a common operation, e.g., when they each take a different stance on a proposition will be addressed later in this section.

3.1. Sharing Information with a Propositional Interlingua

If specialists use different data structures from other specialists, how can they share information? In AMs, this is accomplished through a specialist-neutral propositional interlingua that all specialists must be able to translate into their own. Working out the details of this interlingua is one of the objectives of this project, however preliminary results and other work suggest that a simple first-order propositional language might suffice. The reason for this is that the interlingua is used purely for communication, not inference. Note that an interlingua of FOL propositions does not commit the specialists or any other part of the project to logical techniques. Figure 3 illustrates two specialists using very different internal representations about the same object, but translating it the same interlingua.

3.2. Focus of Attention

When do specialists share information? For several computational reasons, as well as an analogy with human visual attention ([8–10]), specialists in AMs all focus on and share information about a single proposition at a time. This minimizes the chance of a specialist making an inference based on an assumption another specialist knows is false. It also increases the chance that an inference by one specialist will be incorporated into other specialists' inference as soon as possible.

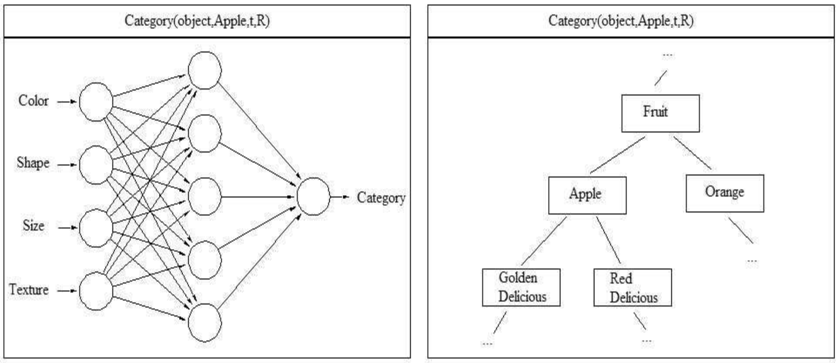


Figure 3. Two specialists using different internal representations but able to share information using a propositional interlingua. A classification specialist (left) stores information about categories in the connections of a neural network. An ontology specialist (right) stores information about categories in a graph. Both can translate this information from and into the same interlingua.

To share information with each other, the specialists implement the following functions:

- `ForwardInferences(P,TV)`. Given a new proposition `P` and its truth value, return a set of truth values for propositions which can now be inferred.
- `Subgoals(P)`. Return a set of propositions whose truth value would help determine `P`'s truth value.
- `Ground(P)`. Return a set of propositions which are a grounding of `P`.
- `SimilarityMatches(P)`. Return a set of propositions representing similarities supported by `P`. For example, learning that *x* is red (*Red(x)*) might lead to the inference that *x* is similar to *y* (*Similar(x,y)*).

3.3. Selecting Attention

The best way to implement a focus manager is a topic this project aims to explore. However, a very simple scheme based on a "focus queue" will help illustrate this approach. Preliminary work demonstrates that even such a simple method can generate important results. We will assume, only for now, that the focus manager is a queue of propositions and that at every time step the first propositions on the queue is chosen. Specialists help guide attention by transforming the queue with the following procedure:

- `PrioritizeFocus(Q)`. Returns a queue that augments a version of Q modified to reflect the propositions the specialist wants to focus on.

At every time step in attention machines, all the specialists focus on the same proposition at the same time. Specifically, for an attention machine with focus manager, `FQ`, at every time step:

- The focus manager chooses a proposition, `focus-prop`, to focus on (in this simple case) by taking the first element of `FQ`.

- For every specialist, `S1`. ("All the specialists learn of each other's opinion on the focus of attention.")
- For every specialist, `S2`:
- `S1.Store(focus-prop, OpinionOn(focus-prop,S2)).`
- For each specialist, `S`, request propositions to be focused on:
- `FQ = S.PrioritizeFocus(FQ).`

Thus, at every time step, specialists focus on a single proposition and execute their common operations on that proposition. In other words, the flow of computation in AMs is guided entirely by the choice of focal proposition.

Because the execution of an algorithm can be conceived of as a sequence of attention fixations (during which common operations are executed on the focused proposition), *how attention is selected determines which algorithm is executed.*

Let us now see how simple attention selection strategies can lead to focus traces that correspond to the execution of some key AI algorithms from different subfields. Simple versions of search, rejection sampling and case-based reasoning can be implemented with the following simple attention control strategies, embodied in the PrioritizeFocus procedure of a specialist, S.

- *Search*. When none of the specialists have any opinion on proposition, `R(x,y,w)`, or when those opinions conflict:
  - o `PrioritizeFocus(FQ)` returns `addFront(R(x,y,w1), addFront(R(x,y,w0),FQ))`, where `addFront(X,Q)` returns a queue that is identical to `Q` with `X` added to the front, `w0` is the possible world where `R` does not hold over `x` and `y` and `w1` is the world where it does.
- *Rejection sampling*. When `S` believes that `R(x,y)` is *p/q* times more likely (for example by keeping track of the relative number of worlds in which `R(x,y)` is true):
  - o `PrioritizeFocus(FQ)` returns `addFrontN(R(x,y,w1), addFrontN(R(x,y,w2), FQ, q), p)`, where `addFrontN(X,Q,N)` returns a queue identical to `Q` with N copies of `X` attached to the front of it and where w1 and w2 are as above. (In English, "focus on the world where `R(x,y,w1)` is true *p/q* times more often than the world where it is false.")
- *Truth maintenance*. When `S` changes its opinion on a proposition `P`:
  - o `PrioritizeFocus(FQ)` returns `addFront(P,FQ)`. ("If you change your opinion on `P`, have all the specialists focus on `P` so they know your new opinion and can revise inferences they made based on that opinion on `P`.")
- *Case-based reasoning*. When having proposition `P` be true is a goal:
  - o `PrioritizeFocus(FQ)` returns `A,R1 … Rn`, where
    - ▪ `S` has retrieved a proposition `A'` representing an action that achieved `P'` such that `Similar(A',A)` and `Similar(P,P')` and
    - ▪ `ri` are propositions of the form `Similar(ri,ri',Wr)`, where `ri'` are participants in `A'` and `A` are participants in `A`. ("If `P` is a

goal, retrieve actions that have achieved similar goals in the past and find analogues for those participants in the present.")

This illustrates how the algorithms from different subfields can be implemented using the same basic common operations.

3.4. Resolving Conflicts Among Specialists and Inference Strategies

How do AMs choose from among the several focus strategies described in the last section? In practice, this problem is less severe than it seems at first glance. A recurrent theme among the algorithms used to illustrate the common operation principle is that they choose which common operation to execute in response to *metacognitive problems:*

1. Search algorithms choose possible worlds by assuming the truth of propositions that are unknown. If the truth value of a proposition is known, it is considered fixed and worlds with that proposition having the opposite value are not explored. In the case of constraint-based search algorithms the metacogntive problem is ignorance, the truth value of the proposition is not known. In the case of planning-based search, the ignorance is often based on conflict, since there is more than one possible next action to take or explore.
2. Stochastic simulation algorithms also choose worlds to explore based on unknown values of a state variable, but the level of ignorance is somewhat reduced since a probability distribution for the state variable is known. This leads to a somewhat different exploration strategy exploring worlds with more likely truth values more often than those with less likely truth values.
3. Resolution-based logic theorem proving algorithms are also driven by ignorance. If the truth value of a literal was known, then an algorithm could simply retrieve it and there would be no need to search for a proof.
4. Case-based reasoning algorithms are also often driven by ignorance and differ from search and simulation algorithms by how they react to the ignorance. Instead of exploring possible worlds where different actions are taken, they retrieve similar solutions to similar problems and try to adapt them.

Thus, case-based reasoning, search and stochastic simulation are different ways of addressing three different kinds of *metacognitive problems*. This is called the *cognitive self-regulation principle*. Stochastic simulation deals with metacognitive problems where there is some information about the likelihood of different solutions to it working. Case-based reasoning deals with metacognitive problems where there are past solutions to similar problems. Search deals with the case where very little is known about the specific metacognitive problem except the basic knowledge about the domain that makes it possible to search for a solution.

Thus, the different algorithms deal with different kinds of metacognitive problems and will therefore conflict with each other rarely. In the current scheme, such rare conflicts are resolved by the order in which specialists modify the focus queue. One specialist can modify the queue and there is nothing to prevent the next specialist which modifies it to undo the modifications of the first specialist. In preliminary work, this has not yet caused any problems, though as implemented AMs and the problems they address become more complex, better methods of managing focus will be needed.

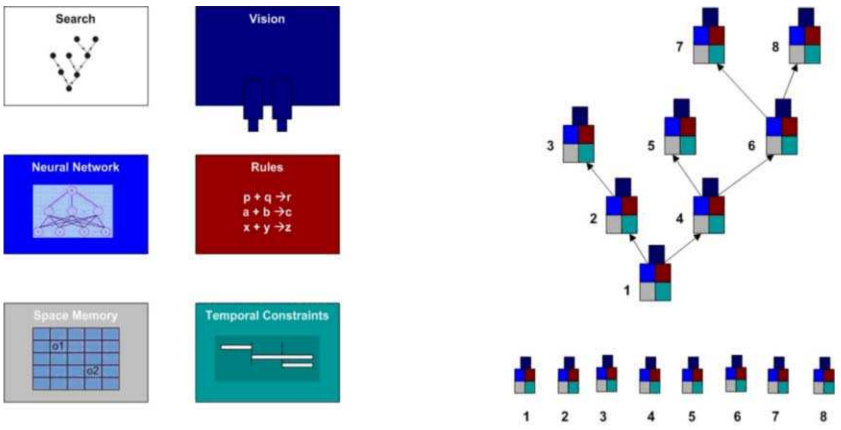


Figure 4. Purely modular integration compared to an attention machine hybrid.

3.5. Extending the Focus Manager

The above discussion demonstrates how AMs with even a very simple focus management scheme (ordering propositions in a queue) can enable extensive hybridization. Queues, however, are almost certainly not a viable mechanisms for achieving the long-term goals of this project. Thus, part of every phrase of this research will be to explore ways of guiding the attention of specialists. In addition to focus management schemes that are motivated by a formal or empirical analysis of particular problems or domains, we will study how focus management can be learned.

One idea is to use state-action-reward reinforcement learning algorithms to automatically generate focus management schemes. The actions are the choice of attention fixation. The state will be both the state of the environment and the goals and subgoals of the AM. The reinforcement signal will be a combination of measures such as how many propositions are in doubt and how many goals are (close to being) achieved. Since in this framework the choice of attention fixation amounts to the choice of algorithm to execute, this approach can show how different kinds of AI algorithms are ways of optimizing cognitive effectiveness on different kinds of problems.

3.6. Summary: Two Ways of Achieving Hybrids in AMs

We have described two ways to make hybrids of algorithms in AMs. The first is to execute a sequence of attention fixations, i.e., a focus trace, that is the combination of focus traces from individual algorithms. The key insight that makes this possible is that many algorithms from very different branches of AI, including case-based reasoning, logic-theorem proving and search can be implemented through sequences of common operations. Figure 2 illustrated this kind of hybridization.

The second way to achieve hybridization is by enabling modules based on many different computational methods to execute common operations. Every algorithm that is executed as a sequence of attention fixations is integrated with each of the algorithms inside of the specialist.

Figure 4 illustrates this kind of integration. On the left, search is implemented in a purely modular system where it is isolated from other computational mechanisms. On

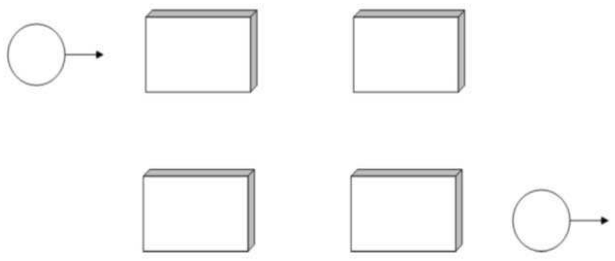


Figure 5. A physical reasoning problem. Is the object that rolls behind the first occluder the same as the object that rolled out of the second.

the right, search is implemented as a sequence of attention fixations that each involves *all* the modules. Thus, every data structure and algorithm inside the modules is integrated with every step of search.

4. Results so Far

Preliminary results with AMs are promising. They demonstrate that AMs can achieve quantifiable improvements over non-hybrid approaches and that they can enable new functionality in real-world systems. The most extensive AM implemented yet is a physical reasoning system for (real and simulated) robots. This section describes this system in relatively more detail as an illustration of how to implement and evaluate AMs. Some other preliminary results are also briefly presented.

4.1. Physical Reasoning

Figure 5 illustrates a (deceptively) simple physical reasoning problem. An object passes behind an occluder; a similar object emerges from another occluder. Are they the same object? The problem of making such inferences for arbitrary configurations and behaviors of objects includes the computational challenges mentioned in the first section. There is *incomplete information*; the state of the world in some locations is hidden from the view of the robot. It is an *open world*; there may be objects hidden behind occluders that are not perceived but which can still affect the outcome of events. The “size” of the problem is very large; even if we idealize space and time into a 100x100x100 grid with 100 possible temporal intervals, there are 10^8 possible place-time pairs that might or might not have an object in them. It is *a dynamic environment*: since objects can move independently of the robot, inferences and plans can be invalidated before they are even completely formulated.

This seemingly simple problem thus contains many of the elements which make AI difficult. It therefore causes difficulties for many existing approaches.

SAT search. SAT search algorithms must represent this domain using a set of propositional constraints. The constraint that objects trace spatiotemporally continuous paths, for example, can be expressed with the following first-order conditional: *Location(o,p1,t1) ^ Location(o,p2,t2) ^ SucceedingTimeStep(t1,t2) → Adjacent(p1,p2)*. For O objects, an NxNxN grid and T time steps, the number of propositional constraints this compiles into is $N^6 \times T \times O$. In a 100x100x100 grid, with 10 objects and 100 temporal intervals, this is a trillion propositional constraints.

Bayesian Networks. Bayesian networks are also propositional and generally have similar problems as SAT as the number of state variables increase.

Logic. Identity (i.e., equality) is a challenge for logic programming systems. For example, for each two-place predicate, a clause such as the following is required: $x1 = x2 \wedge y1 = y2 \wedge R(x1,y1) \rightarrow R(x2,y2)$. A backward-chaining theorem prover operating in a world with 100 objects would have 10,000 possible resolvants to check for each equality literal in a clause such as this.

Also, the closed-world assumption which logical, Bayesian and SAT algorithms typically make does not hold in this domain requiring substantial modifications to all of them. (Though, see (Milch, 2005) for some resent work in open-world probabilistic inference).

4.1.1. How S6 Deals with Computational Challenges to HLI

The following briefly describes several elements of the design of an AM called S6 for conducting physical reasoning.

Open world, big state space. Graphical models, SAT solvers and many kinds of planners is that they instantiate every state variable. In S6, state variables (e.g., representing objects, locations, colors, etc.) are not explicitly represented by specialists until they are focused on. Also, inference is not conducted through a brute-force search or Monte Carlo simulation of the state space. Inference is instead lazy. When a specialist can make a forward inference from a proposition that is focused on it makes it.

Identity. S6 only considers identities between objects that are suggested by a neural network. The network takes as input the properties of an object and outputs potential objects that might be identical to the input object. This greatly reduces the length of inference.

Reactivity. Since it is implemented in an AM, every step of search in S6 is conducted with an attention fixation that involves all of the specialist. This includes specialists that encapsulate sensors. Thus, every step of inference (i.e., every attention fixation) can be informed by new information from a change world. Further, if a perceptual specialist receives information that contradicts something a specialist previously believed, it will request that this new information is focused on. When the new information is focused on, specialists will redo forward inference, retract invalidated inferences and make new inferences based on the information. Thus, having every step of an algorithm implemented as focus of attention of specialists that include perceptual abilities enables inference to react and adjust to new or changed information the moment it is sensed.

4.1.2. Preliminary Evaluation of Physical Reasoner

Several preliminary results attained with S6 suggest hybrid algorithms implemented in AMs can achieve significant improvements over algorithms executing individually, especially on problems with an open world, time, identity, a dynamic environment and noisy sensors.

The first result is that hybrid algorithms in S6 exists and makes inferences in domains where individual algorithms have trouble. For example, a subset of the physical reasoning problem S6 solves was coded in the Discrete Event Calculus ([11]) which uses SAT solvers to solve problems in the event calculus. For an 8x8x8 world with 8 time steps, the computer's available memory (450MB) was exhausted. S6 solved these problems easily. These comparisons need to be conducted with probabilistic inference systems and logic-theorem provers.

These results, of course, come at the cost of soundness and completeness. We suspect that this is a necessary tradeoff and therefore will compare inferences S6 makes against human performance in addition to normative standards. In preliminary work, S6 has already been attained success on many problems in the human physical reasoning literature (e.g., [12]).

S6 demonstrates that hybridization can lead to significant increases in efficiency. For example, as describe above, S6 uses a neural network specialist to suggest equalities. Only these equalities are considered by S6. When this neural network is disabled and search is run alone, S6, becomes extremely slow. The effect has been so dramatic (i.e., several orders of magnitude), that we have not carefully measured it yet, although this is a priority for the near future.

Finally, S6 has been used to design a robotic framework ([13]) for addressing the tension between the apparent rigidity and inflexibility of the sense-plan-act loop of traditional planning algorithms and the autonomy and reactivity required of real-world robots. This framework used specialists to implement reactive subsystems and used a simple attention management scheme to marshal these reactive components to produce deliberate planning and reasoning. The framework was used to create robots that executed traditional planning and reasoning algorithms (e.g., search, rule-matching, means-ends analysis) and could, the minute it was perceived, use changed or updated information about the world to revise past inferences.

4.2. Other Results

Cassimatis (2003) described a system called, Polylog, that implements a logic programming system based on nonlogical data structures and algorithms. Polylog enables specialists based on any data structure or algorithm (for example, relational databases, numerical methods or neural networks) to be integrated in one system and return sound and complete answers to queries. Polylog implemented theorem proving using a focus queue and an attention controls strategy that iteratively made subgoals and grounded propositions. This work suggests that in some cases the tensions between the formal completeness of logic programming and the computational efficiency of specialized data structures and algorithms could be resolved using AMs.

None of this previous work implements the bulk of the ideas described earlier. The attention management schemes were preliminary and simple and many different computational methods were neglected (e.g., graphical models and POMDPs.) Nevertheless, this work demonstrates that the approach of creating hybrid algorithms by implementing them using common operations implemented by modules based on different representations enables a greater degree of robustness.

5. Conclusions

The goal of the work reported in this chapter has been to advance the ability of artificial systems to exhibit human-level intelligence by providing a framework for implementing adaptive algorithmic hybrids. By enabling algorithms to be executed as sequences of attention fixations that are executed using the same set of common functions, it is possible to integrate algorithms from many different subfields of artificial intelligence. This lets the strengths of each algorithm to compensate for the weaknesses of others so that the total system exhibits more intelligence than had previously been possible.

References

[1] P. Agre and D. Chapman, "What are plans for?," Journal for Robotics and Autonomous Systems, vol. 6, pp. 17–34, 1990.
[2] R. A. Brooks, "Intelligence without representation," Artificial Intelligence, vol. 47, pp. 139–159, 1991.
[3] H. H. Hoos and T. Stützle, "SATLIB: An Online Resource for Research on SAT," in SAT 2000, I. P. Gent, H. v. Maaren, and T. Walsh, Eds.: IOS Press, 2002, pp. 283–292.
[4] H. Kautz and B. Selman, "Unifying SAT-based and Graph-based Planning," presented at IJCAI-99, 1999.
[5] J. Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. San Mateo, CA: Morgan Kaufmann, 1988.
[6] J. Gu, "Efficient Local Search for Very Large-Scale Satisfiability Problems," SIGART Bulletin, vol. 3, pp. 8–12, 1992.
[7] N. L. Cassimatis, "A Framework for Answering Queries using Multiple Representation and Inference Technique," presented at 10th International Workshop on Knowledge Representation meets Databases., 2003.
[8] A. M. Treisman and G. Gelade, "A feature integration theory of attention," Cognitive Psychology, vol. 12, pp. 97–136, 1980.
[9] B. J. Baars, A Cognitive Theory of Consciousness. Cambridge: Cambridge University Press, 1988.
[10] J. R. Stroop, "Studies of interference in serial verbal reactions," Journal of Experimental Psychology: General, pp. 622–643, 1935.
[11] E. T. Mueller and G. Sutcliffe, "Reasoning in the event calculus using first-order automated theorem proving," presented at Eighteenth International Florida Artificial Intelligence Research Society Conference, 2005.
[12] E. S. Spelke, "Principles of Object Perception," Cognitive Science, vol. 14, pp. 29–56, 1990.
[13] N. L. Cassimatis, J. G. Trafton, M. Bugajska, and A. C. Schultz, "Integrating Cognition, Perception, and Action through Mental Simulation in Robots," Robotics and Autonomous Systems, vol. 49, pp. 13–23, 2004.

Advances in Artificial General Intelligence: Concepts, Architectures and Algorithms
B. Goertzel and P. Wang (Eds.)
IOS Press, 2007

© 2007 The authors and IOS Press. All rights reserved.

Cognitive Map Dimensions of the Human Value System Extracted from Natural Language

Alexei V. SAMSONOVICH[a] and Giorgio A. ASCOLI[b]
[a]*Krasnow Institute for Advanced Study, George Mason University, Fairfax, VA 22030*
[b]*Krasnow Institute for Advanced Study and Psychology Department, George Mason University, Fairfax, VA 22030-4444*

Abstract. The notion of a human value system can be quantified as a cognitive map, the dimensions of which capture the semantics of concepts and the associated values. This can be done, if one knows (i) how to define the dimensions of the map, and (ii) how to allocate concepts in those dimensions. Regarding the first question, experimental studies with linguistic material using psychometrics have revealed that valence, arousal and dominance are primary dimensions characterizing human values. The same or similar dimensions are used in popular models of emotions and affects. In these studies, the choice of principal dimensions, as well as scoring concepts, was based on subjective reports or psycho-physiological measurements. Can a cognitive map of human values be constructed without testing human subjects? Here we show that the answer is positive, using generally available dictionaries of synonyms and antonyms. By applying a simple statistical-mechanic model to English and French dictionaries, we constructed multidimensional cognitive maps that capture the semantics of words. We calculated the principal dimensions of the resultant maps and found their semantics consistent across two languages as well as with previously known main cognitive dimensions. These results suggest that the linguistically derived cognitive map of the human value system is language-invariant and, being closely related to psychometrically derived maps, is likely to reflect fundamental aspects of the human mind.

Keywords. Cognitive architectures, affective dimensions, psychometric.

Introduction

Neuromorphic cognitive map is a functional unit that plays a central role in the Biologically Inspired Cognitive Architecture developed at George Mason University (BICA-GMU) by our research team [1-4]. The notion of a cognitive map, however, is not limited to the field of artificial general intelligence (AGI). The term “cognitive map” had been used in cognitive sciences for several decades with various meanings [5, 6]. The modern notion of a cognitive map was introduced by O’Keefe and Nadel [7] based on the hippocampal place cell phenomenon discovered at that time [8]. This notion was subsequently extended to include cognitive mapping of non-spatial features of contexts and paradigms, based on the spatial analogy [e.g., 9-12].

In the present work, a *cognitive map* is understood as a mapping from a set of cognitive representations (e.g. concepts, words) to an abstract continuous metric space, such that semantic relations among representations are reflected in geometric relations in the indexing space. This intuitive definition unfolds as follows. In a spatial cognitive map, the metrics are proportional to the perceived distances between associated landmarks in the physical world. In this case the cognitive map is essentially a model of perceived space [7]. Similarly, a temporal cognitive map, if it were found in the brain as a separate functional unit, would be a model of a perceived timeline. Another example of a cognitive map is a color space, in which locations correspond to perceived colors [13].

Speaking more generally, we distinguish various kinds of cognitive maps (Figure 1), based on the semantics they represent (logic, values, feelings, qualia) and on the representation systems they map (e.g., one may distinguish contextual and conceptual cognitive maps). While the idea of mapping representations of concepts onto an abstract space is not new [14], cognitive maps beyond spatial and temporal dimensions remain unexplored terrain in cognitive neurosciences. How to design a cognitive map: its topology, geometry, and the associated semantics? How to allocate representations on a map? Can this be done objectively, and/or from the first principles?

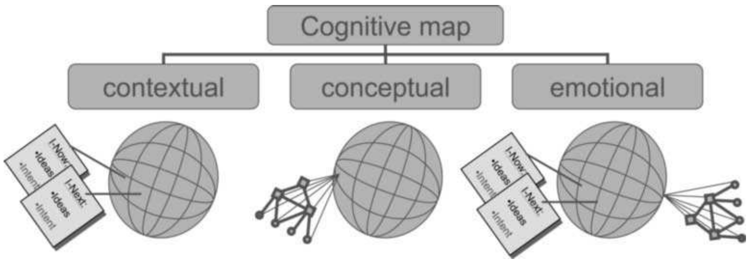


Figure 1. Cognitive maps are abstract metric spaces that reflect semantics of associated symbolic representations. Different kinds of cognitive maps may represent different aspects of semantics and/or map different kinds of representations.

Here we focus on a particular kind of cognitive maps: *conceptual value maps*. We believe that the notion of a human value system can be made more precise and more useful when it is represented with a cognitive map, the dimensions of which capture the values of concepts. This kind of a map can be constructed, if one knows how to define the dimensions of the map and how to allocate concepts in them.

The idea of the approach pursued in the present study is to use linguistic corpora as a source of data about the semantics of concepts (represented in this case by words). The hope is that self-organization may help us find the principal dimensions and to allocate concepts automatically. We expect that the problems of cognitive map creation can be solved in this case using available linguistic data. Therefore, we select a dictionary of words as our study material, keeping in mind that words represent concepts, and concepts are associated with values.

Simple as it is, the idea of applying the notion of a cognitive map to linguistic corpora appears to be unexplored, while multidimensional metrics were used to characterize semantics of words, concepts and feelings in many cognitive studies. Examples include theoretical models of emotions [e.g., 15, 16] that go along with experimentally derived psychometric dimensions of words [17, 18] and partially overlap with abstract studies of "quality dimensions" [14]. All of the above influenced modern cognitive architecture designs that make use of principal cognitive dimensions (see, e.g., Adams, this volume). In the aforementioned experimental studies, the choice of principal dimensions, as well as related scoring of concepts, was based on subjective reports or psycho-physiological measurements. Can a cognitive map of human values be constructed without testing human subjects? Here we show that the answer is positive, using generally available dictionaries of synonyms and antonyms.

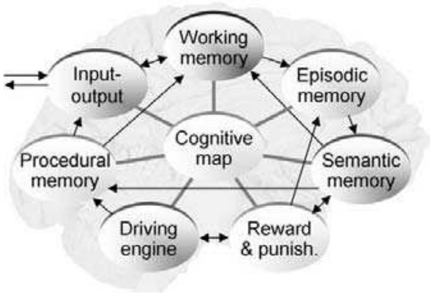


Figure 2. A bird view of BICA-GMU, as described in [2].

The present study of cognitive maps is best framed in the context of our cognitive architecture design: BICA-GMU (Figure 2). Information processing in BICA-GMU occurs at a higher symbolic level, based on new building blocks called “a schema” and “a mental state” [2]. Intuitively, the notion of a schema can be associated with that of a concept, while the notion of a mental state can be associated with that of a context. Multiple instances of schemas and mental states fill in the three main memory systems in BICA-GMU: working, semantic and episodic memory. When new information comes to the system through the input-output buffer, it gets represented by instances of schemas. This is done with the help of procedural memory that "knows" what schemas should be used for each given kind of input. The rest of information processing does not have a pre-defined solution and may involve search of the entire semantic and/or episodic memories at each step. Thus, filtering of the exploding tree of possibilities becomes vital for successful operation of BICA-GMU in practical scenarios. This filtering is one of the main functions of neuromorphic cognitive maps in BICA-GMU. In general, it can be understood as a task to suggest a preferred choice of a schema that will be used by the architecture at the next step. Filtering by a cognitive map also constrains the semantics of admissible schemas to a narrow domain in the cognitive space. This mechanism could be used, e.g., in analogy search or in classification of memories. Another aspect of the same cognitive map function is evaluation (how good, how plausible, how exciting, etc.) of a given concept. In this sense, cognitive maps provide an otherwise undefined “metric system” in the field. While the cognitive map is expected to develop its metrics through self-organization, the primary values for a limited number of selected concepts need to be provided by an external source. In the case of BICA-GMU they are provided by the reward and punishment system (Figure 2).

1. Materials and Methods

1.1. Linguistic Corpora

The study presented here was conducted using two linguistic corpora: dictionaries of synonyms and antonyms available as parts of the thesaurus in Microsoft Word 2003 (MS Word). The two corpora apparently have independent origin[1] and different characteristics. The total size of each of them is above 200,000 entries. The core dictionaries used in this study and the corresponding matrices *W* of synonym-antonym relations were extracted automatically following the algorithm described below.

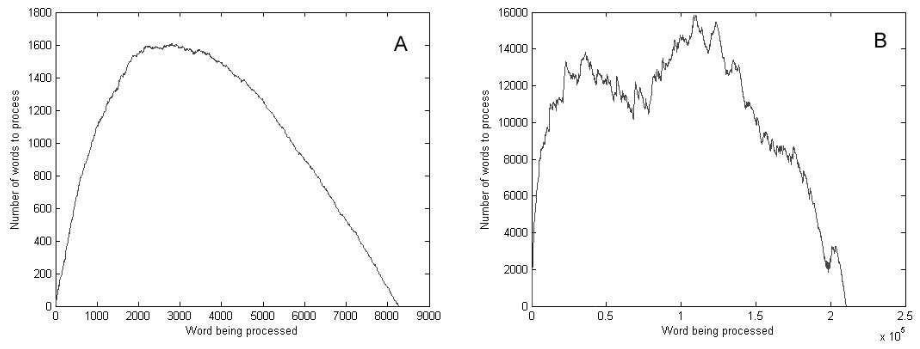


Figure 3. Extraction of the dictionary from MS Word. **A:** English, **B:** French. Abscissa: the word that is being processed. Ordinate: the number of unprocessed words in the retrieved list at the current step.

1.1.1. Extraction of the Core Dictionary

The following algorithm was used to extract a core dictionary from MS Word.

Step 1: Start with one word in the dictionary. Take the next word from the dictionary. Retrieve its synonyms and antonyms from MS Word. Merge them into the dictionary (avoid repetitions). Repeat until the retrieved dictionary is processed.

Step 2: Eliminate duplicates that were not detected during Step 1. Recursively remove all words with less than two connections (see Figure 4 A). The remainder by definition constitutes the "core" dictionary. Symmetrize the relation matrix *W* by making all synonym and antonym links bi-directional[2].

The starting word for English was "first", for French "premier". The resultant sets of words never changed by more than a few words when we tried different starting words.

[1] English thesaurus was developed for Microsoft by Bloomsbury Publishing, Plc. French thesaurus is copyrighted by SYNAPSE Development, Toulouse, France.

[2] Symmetrization is necessary for the energy function (*) to be Hermitean, in which case the relaxation process (**) converges to a fixed point rather than a cycle.

1.1.2. Characteristics of the Core Dictionaries

The extracted English core has 8,236 words. An average word in it has 3.0 synonyms (1.8 before symmetrization) and 1.4 antonyms (0.8 before symmetrization). The extracted French core has 87,811 words. An average word in it has 6.4 synonyms (3.9 before symmetrization) and 7.5 antonyms (3.9 before symmetrization). The total average number of connections (degree) per word is 4.3 for English core and 14.0 for French core. In each case, the extracted core is a small part of the entire thesaurus.

The graph of synonym-antonym links for the English core is nearly scale-free [19] (Figure 4, cf. [20]), but cannot be used as a cognitive map. For example, considering the graph of English synonyms only, one can see that distances on the graph (measured in the number of links of the shortest path) are semantically misleading:

- Average distance between words = 6.7
- Distance (true, false) = 8
- Distance (big, small) = 5
- Distance (happy, satisfaction) = 7
- Distance (cloth, dress) = 5

This happens because only very few of all synonyms and antonyms of a given word are actually listed in the corpus (the graph is very sparse).

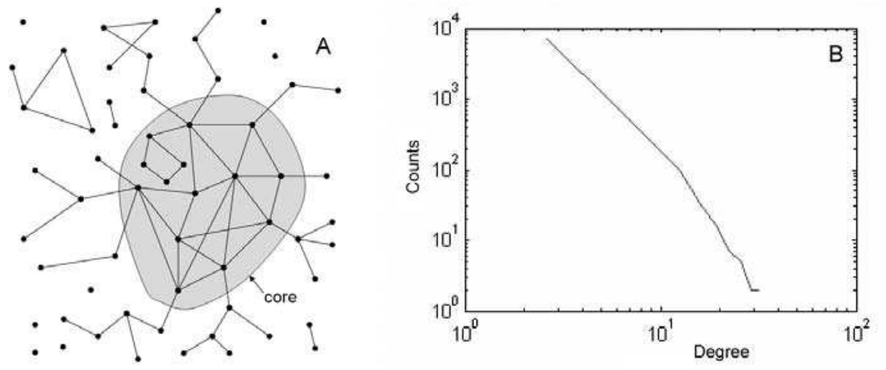


Figure 4. Post-processing of the extracted dictionary. **A:** The notion of a core dictionary. Only nodes that have at least two connections to other core nodes are left in the core. Links are bi-directional and could be synonym-synonym or antonym-antonym connections. **B:** Scaling law analysis [19] shows that the extracted English core dictionary forms a nearly scale-free graph, which is typical for word-association networks [20].

1.2. Statistical-Physics-Inspired Approach to Self-Organization of a Cognitive Map

The idea of our approach to constructing a cognitive map by self-organization is to represent the evolving cognitive map by a statistical mechanical model and to find its ground state that is expected to capture the semantics. Therefore, the heart of our method is the following algorithm, which was designed through trial and error[3].

1. Randomly allocate N words as "particles" (vectors) in R100 in a unit ball.
2. Arrange for attraction between synonyms and repulsion among antonyms by defining an energy function of the system based on the relation matrix W:

$$H(\mathbf{x}) = -\frac{1}{2}\sum_{i,j=1}^{N} W_{ij}\mathbf{x}_i \cdot \mathbf{x}_j + \frac{1}{4}\sum_{i=1}^{N}|\mathbf{x}_i|^4, \quad \mathbf{x} \in \mathrm{R}^N \oplus \mathrm{R}^{100}. \tag{1}$$

3. Simulate thermodynamical relaxation of the system to its ground state (106 iterations) based on the following stochastic equation (η is a Gaussian noise):

$$\dot{\mathbf{x}}_i = -\frac{\partial H}{\partial \mathbf{x}_i} + \boldsymbol{\eta}_i(t), \quad \left\langle \boldsymbol{\eta}(t)^2 \right\rangle \to 0. \tag{2}$$

4. Rotate the resultant distribution in R^{100} to its principal components (PCs).
5. Identify semantics of the coordinates (PCs) by sorting words along them.

The symmetric relation matrix *W* in (1) has "+1" entries for pairs of synonyms and "–1" entries for pairs of antonyms, all other matrix elements are equal to zero. During the construction of *W*, different forms of the same word were treated as synonyms. Convergence of (2) to a ground state was assessed by measuring the maximal displacement of any "particle" in one step of simulated dynamics.

1.3. Psychometric Data Used in This Study

In the analysis of our results we used the Affective Norms for English Words (ANEW) database [21] developed by the Center for the Study of Emotion and Attention (CSEA) at the University of Florida. This database contains 1,034 affective English words. The ANEW database was created using the Self-Assessment Manikin to acquire ratings of *pleasure*, *arousal*, and *dominance*. Each rating scale in ANEW runs from 1 to 9, with a rating of 1 indicating a low value (low pleasure, low arousal, low dominance) and 9 indicating a high value on each dimension. The ANEW database was kindly provided by Dr. Margaret M. Bradley (University of Florida, CSEA).

1.4. Software and Hardware

Algorithms were implemented using XEmacs and GNU C on Dell Optiplex GX620 running Fedora Core 5 Linux. Data preparation and analysis were performed using Microsoft Office 2003 Professional Enterprise Edition on Dell Optiplex GX620 running Windows XP Pro, also using Octave and Matlab 7.0.

2. Results

In all our numerical experiments, 10^6 iterations (and 10^5 iterations in most cases) were sufficient for convergence of the system to its ground state (assessed as described above). In the numerical implementation of (2), the time step was $\Delta t = 0.001$, the initial

[3] We tried various topological embeddings and constraints: a ball, a sphere, a torus, an ellipsoid, with the dimensionality ranging from 1 to 100, and selected an optimum.

noise standard deviation $<\eta^2>^{1/2}$ was 0.5, and its value decreased inversely proportionally with time. Our key findings are presented in Figures 5-7 and Tables 1-2.

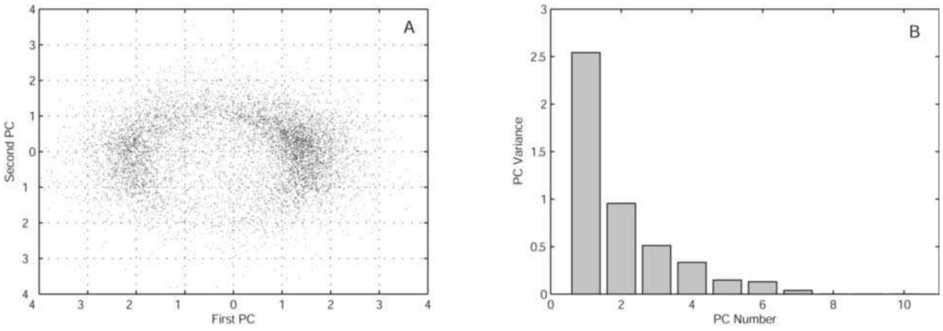


Figure 5. The English core in its “ground state”. **A:** The “banana shape” of a final distribution of English words (visible after rotation to the main PC coordinates). The two fuzzy clusters and the corresponding horizontal dimension separate positive and negative values (*valence*). The vertical axis corresponds to another cognitive dimension: “calming vs. exciting” (*arousal*). D=100, H(x) is given by (*) in Section 1.2. **B:** Variances of the first ten PCs (Table 1). PCs are sorted by their variance.

Table 1. Sorted lists for the English core in a ground state. 10 PCs, 5+5 elements of each list. The order number and the variance of each PC are given in the left column.

PC # & Variance	Starting from the one end of the list:	Starting from the other end of the list:
1: 2.54	increase, well, rise, support, accept…	drop, lose, dull, break, poor…
2: 0.95	calm, easy, soft, gentle, relaxed…	difficult, harsh, hard, trouble, twist…
3: 0.51	start, open, fresh, begin, release…	close, delay, end, finish, halt…
4: 0.33	thin, edge, use, length, wet…	center, save, deep, dry, middle…
5: 0.15	essential, need, poverty, basic, necessary…	back, surplus, later, wealth, unnecessary…
6: 0.13	pull, private, receive, owe, keep…	dig, push, channel, ditch, national…
7: 0.04	over, top, on top of, above, impose…	base, under, below, underneath, beneath…
8: 5.6e-8	old, mature, adult, aged, previous…	young, child, dig, immature, new…
9: 1.7e-9	normally, carefully, modestly, frequently, often…	unusually, extremely, rarely, strangely, carelessly…
10: 6e-10	personally, carefully, for myself, amazingly, thoughtfully…	universally, carelessly, normally, generally, usually…

The shape of resultant distributions (Figures 5, 6) was found independent of the initial conditions and the realization of the stochastic noise η, which was sampled from a normal distribution. Interestingly, the geometric properties of shapes of the final distributions for the two languages are similar (cp. Figure 5 A and Figure 6 A): these are bimodal, “banana-shape” distributions, each exhibiting two dense clusters connected by a curved “neck”, surrounded by a diffuse “fringe”.

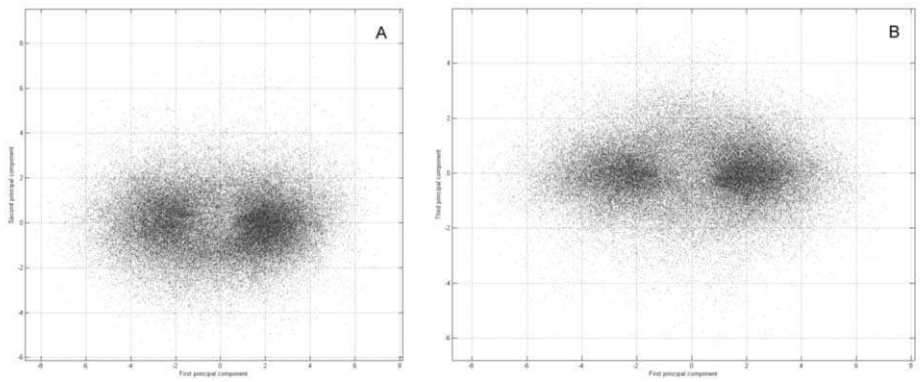


Figure 6. The French core in its “ground state”. **A:** PC #1 vs. PC #2. The “banana shape” distribution resembles that of Figure 5 A. **B:** PC #1 vs. PC #3. Like in the case of Figure 5, the distribution is bimodal with two clusters corresponding to positive and negative values (as found by examining the cluster elements).

Another interesting detail is that the PC amplitude is quickly decaying with the PC number (Figure 5 B and Table 1, left column), which may be a consequence of the fact that the matrix *W* is very sparse – or an indication that there are indeed very few cognitive dimensions in the corpus. Consistently with the first interpretation, the decay is slower, yet also very strong, for the French core (not shown here).

The findings of similarities in geometry extend to the semantics of distributions: Tables 1, 2 present the end-portions of sorted lists of words (words were sorted by their PC scores). Table 1 suggests that there are definite semantics associated with each of the first ten PCs: an intuitive impression that is difficult to quantify. Table 2 was prepared as follows. Each French sorted list was subjected to automated translation into English, after which duplicate words in it were removed, and then both lists within each row were truncated to an equal size. Observed semantic correlations are significant.

Table 2. Top portions of sorted lists for each PC: top three dimensions for two languages. Words that are common across cells within each row are typed in boldface.

PC #	English: 8,236 words total core size	French: translated (87,811 words total core size)
1	**increase**, well, rise, support, **accept**, clear, improve, right, continue, direct, good, make, respect, honor, **happy**, secure, order, understanding, **fix**, power, bright, present, definite…	**happy**, agreement, stable, joined together, delighted, approve, net, some, honest, rich, added, **increased**, pleasant, sincere, union, frank, **fix**, favor, praise, optimist, **accept**, abundance, help…
2	**calm**, easy, **soft**, gentle, relaxed, light, ease, **simple**, quiet, soothe, smooth, empty, mild, weak, gently, peaceful, compliant, lenient, pale…	**calm**, modest, discrete, **simple**, subjected, thin, alleviated, softened, flexible, sober, moderate, **soft**, immobility, measured, silence, humble, reserved, simplicity, obeying
3	start, **open**, fresh, begin, **release**, original, new, reveal, speed up, **free**…	**release**, deliver, freedom, yield, **open**, leave, **free**, disencumbered, discovered, dispersion, broad…

2.1. Main Result: Cross-Language Semantic Consistency of the Cognitive Map Structure

Each of the three top portions of paired lists shown in Table 2 has at least three words in common with its counterpart. All words within each row have similar or closely related semantics. Semantic similarities within and across columns of the table seem to be at the same level of strength; however, an objective measure would be necessary to quantify this impression. How can we estimate the statistical significance of co-occurrence of the same words in top portions of two lists in each row of Table 2? Here is one easy way to estimate *p*-values from above. Given the size of the English core, and assuming that each French-to-English translation is a "blind shot" into the English core (null-hypothesis), we can estimate the probability to find one and the same word in top-twelve portions of both lists: $p \sim 2*12*12 / 8{,}236 = 0.035$ (we included the factor 2, because there are two possible ways of aligning the lists with respect to each other[4]). Therefore, the *p*-value of the case of word repetition that we see in Table 2 is smaller than 0.035, at least. In conclusion, we have found significant correlations among sorted lists across languages for each of the three PCs. It is also remarkable that there are no common words shared by any two rows in Table 2.

2.2. Testing the Constructed Cognitive Map: Synonym Selection with a Bias

Our analysis so far was focused on extremities of the emergent cognitive maps. It is natural to ask whether words are organized consistently in the middle of a cognitive map. To address this question, a simple experiment was conducted. The algorithm was the following. (1) Select a word from the core: e.g., "*problem*". (2) List all its synonyms. (3) Sort them along a given dimension: e.g., PC #1. (4) Take one word from the top of the list and one word from the bottom. Below is an example of an outcome.

[4] The "top end" of each English list was in effect selected at random, but the "top end" of the counterpart French list was selected in order to match the semantics of the English list: here we had two possibilities and selected one of them, for each PC.

From the top: Problem → exercise.
From the bottom: Problem → obstacle.

Intuitively, the reader would probably agree that "exercise" is a more positive choice than "obstacle": when I am in a positive mood, I am more likely to take a new encountered problem as an exercise rather than an obstacle. This observation indicates that the cognitive map is consistently organized within itself. How can one quantify the consistency of its internal organization? We address this topic immediately below.

2.3. Analysis by Comparison with Psychometric Data

In order to further validate our findings, we compared our principal dimensions found in the English core against the dimensions of the ANEW dataset: *pleasure*[5], *arousal* and *dominance*. The ANEW list contains 1,034 words, 479 of which were found in the English core. The scatter plot of our PC #1 versus the first dimension of ANEW, which is the mean value of *pleasure*, is represented in Figure 7. The plot shows strong correlation, with similar bimodal distributions in both PC #1 and the ANEW-*pleasure* dimensions. Pearson correlation coefficient $r = 0.70$.

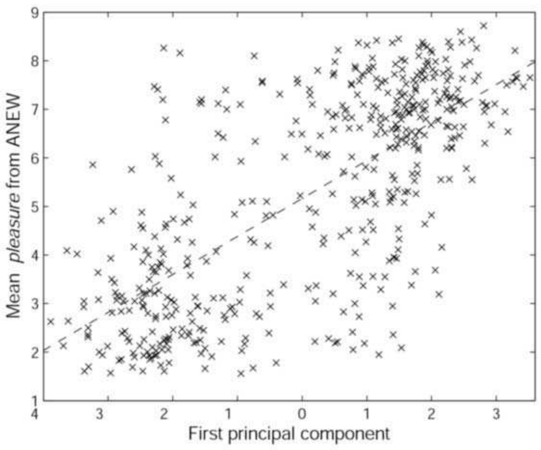


Figure 7. Scatter plot demonstrating strong correlation of PC #1 with the first dimension of ANEW: pleasure. The dashed line is a linear fit. The two clusters ("positive" and "negative") are separated in each dimension.

How can we match PCs with ANEW dimensions? Our correlation analysis shows that PC #1 is the best match (i.e., most highly correlated among all PCs) for ANEW-pleasure, and vice versa ($r = 0.70$, $p = 10^{-70}$). PC #2 is the best match for ANEW-arousal ($r = 0.32$, $p = 10^{-12}$). Finally, ANEW-dominance among all ANEW dimensions

[5] In the ANEW dataset, the first dimension is called "pleasure", while in some studies based on ANEW it is called "valence", consistently with the names of Osgood's dimensions [17].

is the best match for our PC #3 (r = 0.25, p = 10^{-7}); however, PC #1 and ANEW-dominance are correlated stronger (r = 0.64).

Why do arousal and dominance have so low (albeit significant) correlations with the matching PCs? One possible answer is that semantics of ANEW-arousal and ANEW-dominance is different from the semantics of our PC #2 and PC #3. Indeed, in the given subset of 479 words that are common between the English core and ANEW, ANEW dimensions #1 ("pleasure") and #3 ("dominance") are strongly correlated (r = 0.86). On the other hand, our PC #1 and PC #3 are not strongly correlated (r = 0.13), because PCs were calculated using principal component analysis (hence they could be expected to be independent). Debates continue in the literature as to whether *dominance* should be considered an independent dimension [22].

3. Discussion

In the present work, by applying a simple statistical-mechanic model to English and French dictionaries, we constructed multidimensional cognitive maps that capture the semantics of words. We calculated the principal dimensions of the resultant cognitive maps (the PCs) and found their semantics consistent across two languages. In this context it would be interesting to analyze other languages. Our preliminary results of a similar study conducted with a Spanish dictionary of synonyms and antonyms (not reported here) support all our key findings described above, including semantics of the first three PCs.

The principal dimensions that we found appear to be approximately semantically consistent with the previously known dimensions ("affective dimensions" or "Osgood's dimensions") determined psychometrically, in experimental studies with human subjects (e.g. [17, 18] and followers): those studies have revealed that *valence*, *arousal* and *dominance* are the primary dimensions characterizing human values. In this context, it may not be a surprise to learn that modern theoretical models of emotions also use a similar or closely related set of principal dimensions [16, 23]. In the foregoing experimental studies, in contrast with our study, the initial choice of the set of dimensions and respective scoring concepts, was based on subjective reports or psychophysiological measures (SCR, HR, EEG, etc.). In the present work we show that the same or similar dimensions can be found, and a cognitive map of human values can be constructed, without testing human subjects, but using linguistic corpora instead.

At the same time, the match between our PCs and ANEW dimensions labeled by words "arousal" and "dominance" is not perfect. We surmise that semantics of our extracted dimensions cannot be characterized completely by a single word (e.g., "pleasure", "arousal", "dominance") or a pair of antonyms. One may need the entire cognitive map to define semantics of dimensions of this map precisely; however, an approximate definition could be based on a small set of antonym pairs, selected based on their map coordinates, degrees and frequencies. This will be done elsewhere.

Nevertheless, results of comparison with ANEW are surprising. We had no a priori reason to believe that ANEW-dimension #1 (pleasure) and ANEW-dimension #2 (arousal) should correspond to our PC #1 and PC #2. The fact that they do show significant semantic correlations and make best matches with those counterparts is in and by itself intriguing and worth attention. It suggests that the linguistically derived cognitive map dimensions found in this study are not only language-invariant. They appear to be invariant at a broad scale of methodologies, across fields of science, and therefore they are likely to reflect fundamental aspects of human cognition. Extending this logic, one may expect similar results when the same method is applied to other representation systems: corpora, ontologies, databases and indices of various nature, as long as the notions of similarity and contrast can be defined for their elements. The question of whether the same semantics will hold for the first two or three principal dimensions in those cases remains open.

The possibility to construct a complete cognitive map of natural language based on semantic differences and similarities among words opens many important questions. What is the number of independent dimensions of the conceptual value map (cf. [24])? Which of the previous intuitively defined dimensions are orthogonal, and which would be represented as linear combinations of others? What is a canonical choice of a coordinate system, if this notion makes sense, and what would be the semantics of those special coordinates? Would all answers to the above questions be consistent across individuals, social groups, languages – or be individual-specific? Would they be extensible beyond human cognition: to robots and other forms of intelligence? We believe that answering these questions would greatly benefit modern science and technology, because understanding the human value system and the degree of its logical necessity is a key to understanding the laws of cognitive explosion on Earth.

Viewing the findings of the present work in the context of cognitive architecture design, we can imagine an artificial cognitive system that learns or generates new concepts and assigns values to them "on the fly": i.e., in the process of (unsupervised) learning. This may be possible, as soon as the system "knows" how to identify synonyms and antonyms of a new concept among familiar concepts that are already allocated on the cognitive map. Given this strategy, and assuming that the system is capable of cognitive growth[6], we can imagine as the system will gradually develop a personal system of higher values and ideals starting from primitive notions of reward and punishment. This capability could be vital for cognitive systems growing up in social embedding and intended to become human partners.

What else could cognitive maps be used for in a cognitive architecture? Here is an abridged list of their possible functions: filtering of search trees, finding analogies, satisfying semantic constraints, building systems of values for new domains of knowledge, bootstrapping development of higher values and goals, suggesting a reasonable commonsense initiative, classification of memories and strategic retrieval of episodic memories, guidance in imagery and decision making, etc. In addition, linguistic cognitive maps may find applications elsewhere: intuitive Internet search, etc.

In summary, the main finding of this work is the possibility to extract language-invariant dimensions of the human value system from linguistic corpora, using a statistical-mechanic approach. Similar results are expected with other representation systems and databases, including AGI. The principles of map construction, as well as the map itself extracted from the human language, can be used in a general-purpose self-aware cognitive architecture in order to enable its autonomous cognitive growth.

[6] Cognitive growth is understood in modern artificial intelligence as multi-level bootstrapped learning, starting from primitive knowledge and gradually developing higher abstract concepts and goals based on previously learned concepts (e.g., [25]).

Acknowledgments

We are grateful to Dr. Kenneth A. De Jong for valuable discussions and interest to this work. This work is supported by the DARPA IPTO BICA Grant "Integrated Self-Aware Cognitive Architecture".

References

[1] Samsonovich, A. V., & De Jong, K. A. (2003). Meta-cognitive architecture for team agents. In Alterman, R., & Kirsh, D. (Eds.). Proceedings of the 25th Annual Meeting of the Cognitive Science Society (CogSci2003), pp. 1029-1034. Boston, MA: Cognitive Science Society.

[2] Samsonovich, A. V., & De Jong, K. A. (2005). Designing a self-aware neuromorphic hybrid. In Thorisson, K.R., Vilhjalmsson, H., & Marsela, S. (Eds.). AAAI-05 Workshop on Modular Construction of Human-Like Intelligence, Pittsburg, PA, July 10. AAAI Technical Report, vol. WS-05-08, pp. 71- 78. Menlo Park, CA: AAAI Press.

[3] Samsonovich, A. V., Ascoli, G. A., De Jong, K. A., & Coletti, M. A. (2006). Integrated hybrid cognitive architecture for a virtual roboscout. In Beetz, M., Rajan, K, Thielscher, M., & Rusu, R.B. (Eds.). Cognitive Robotics: Papers from the AAAI Workshop. AAAI Technical Report, vol. WS-06-03, pp. 129-134. Menlo Park, CA: AAAI Press.

[4] Samsonovich, A. V. (2006). Biologically inspired cognitive architecture for socially competent agents. In Upal, M. A., & Sun, R. (Eds.). Cognitive Modeling and Agent-Based Social Simulation: Papers from the AAAI Workshop. AAAI Technical Report, vol. WS-06-02, pp. 36-48. Menlo Park, CA: AAAI Press.

[5] Tolman, E. C. (1948). Cognitive maps in rats and man. Psychological Review 55 (4): 189-208.

[6] Downs, R. M., & Stea, D. (1973). Cognitive maps and spatial behavior: Process and products. In Downs, R. M., & Stea, D. (Eds.). Image and Environments, pp. 8-26. Chicago: Aldine.

[7] OKeefe, J., & Nadel, L. (1978). The Hippocampus as a Cognitive Map. New York, NY: Clarendon.

[8] O'Keefe, J., & Dostrovsky, J. (1971). The hippocampus as a spatial map: Preliminary evidence from unit activity in the freely-moving rat. Brain Research 34: 171-175.

[9] Samsonovich, A., & McNaughton, B. L. (1997). Path integration and cognitive mapping in a continuous attractor neural network model. Journal of Neuroscience 17 (15): 5900-5920.

[10] Nadel, L. & Hardt, O. (2004). The spatial brain. Neuropsychology 18 (3): 473-476.

[11] Samsonovich, A. V. & Ascoli, G. A. (2005). A simple neural network model of the hippocampus suggesting its pathfinding role in episodic memory retrieval. Learning & Memory 12 (2): 193-208.

[12] Manns, J. R. & Eichenbaum, H. (2006). Evolution of declarative memory. Hippocampus 16 (9): 795-808.

[13] Palmer, S. E. (1999). Color, consciousness, and the isomorphism constraint. Behavioral and Brain Sciences 22 (6): 923.

[14] Gärdenfors, P. (2004). Conceptual Spaces: The Geometry of Thought. Cambridge, MA: MIT Press.

[15] Oatley, K., & Johnson-Laird, P. N. (1987). Towards a cognitive theory of emotions. Cognition and Emotion 1: 29-50.

[16] Ekman, P., Levenson, R. W., & Friesen, W. V. (1983). Autonomic nervous system activity distinguishes among emotions. Science 221 (4616): 1208-1210

[17] Osgood, C. E., Suci, G. J., & Tannenbaum, P. H. (1957). The Measurement of Meaning. Chicago: University of Illinois Press.

[18] Rubin, D. C. (1980). 51 properties of 125 words: a unit analysis of verbal behavior. Journal of Verbal Learning and Verbal Behavior 19: 736-755.

[19] Barabasi, A.-L., & Bonabeau, E. (2003) Scale-free networks. Scientific American, May 2003: 60-69.

[20] Steyvers, M., & Tenenbaum, J. B. (2005) The large-scale structure of semantic networks: Statistical analyses and a model of semantic growth. Cognitive Science 29 (1): 41-78.

[21] Bradley, M. M., & Lang, P. J. (1999). Affective norms for English words (ANEW): Stimuli, instruction manual and affective ratings. Technical report C-1. Gainesville, FL: University of Florida.

[22] Stamps, A. E. (2005). In search of dominance: The case of the missing dimension. Perceptual and Motor Skills 100 (2) 559-566.

[23] Rolls, E. T. (2006). Emotion Explained. New York: Oxford University Press.

[24] Powers, M. J. (2006) The structure of emotion. Cognition and Emotion 20 (5): 694-713.

[25] Samsonovich, A. V., Ascoli, G. A., & DeJong, K. A. (2006). Human-level psychometrics for cognitive architectures. In Smith L, Sporns O, Yu C., Gasser M., Breazeal C., Deak G., & Weng J. (Eds.).

Proceedings of the Fifth International Conference on Development and Learning ICDL 2006, CD-ROM, ISBN 0-9786456-0-X. Bloomington, IN: Dept. of Psychological and Brain Sciences, Indiana University.

Advances in Artificial General Intelligence: Concepts, Architectures and Algorithms
B. Goertzel and P. Wang (Eds.)
IOS Press, 2007

© 2007 The authors and IOS Press. All rights reserved.

Program Evolution for General Intelligence

Moshe LOOKS (moshe@metacog.org)
Washington University in St. Louis,
Department of Computer Science
and
Science Applications International Corporation (SAIC),
Integrated Intelligence Solutions Operation
and
Novamente LLC

Abstract. A program evolution component is proposed for integrative artificial general intelligence. The system's deployment is intended to be comparable, on Marr's level of computational theory, to evolutionary mechanisms in human thought. The challenges of program evolution are described, along with the requirements for a program evolution system to be competent - solving hard problems quickly, accurately, and reliably. Meta-optimizing semantic evolutionary search (MOSES) is proposed to fulfill these requirements.

1. Background and Motivation

"At every step the design logic of brains is a Darwinian logic: overproduction, variation, competition, selection … it should not come as a surprise that this same logic is also the basis for the normal millisecond-by-millisecond information processing that continues to adapt neural software to the world." *Terrence Deacon* [1]

In David Marr's seminal decomposition, any information-processing system may be understood at three nearly independent levels: (1) *computational theory*, a description of the problems the system attempts to solve; (2) *representations and algorithms*; and (3) *implementation*, the physical instantiation of the system's representations and algorithms [2]. What might the subsystems of human cognition look like on the level of computational theory?[1]

I propose that evolutionary learning be considered as one of these subsystems. Major subfields of both cognitive science and AI are concerned with evolutionary learning processes. One motivator for this concern is an attempt in both fields to augment purely local techniques such as Hebbian (associative) learning with more global methods, which try to make large leaps to find answers far removed from existing knowledge. This is a form of evolutionary learning [4], which Edelman [5] has presented as "Neural Darwinism", and Calvin and Bickerton [6, 7] as the notion of mind as a "Darwin Machine".

It is known that the immune system adapts via a form of evolutionary learning, and Edelman [5] has proposed that the brain does so as well, evolving new "neuronal maps", patterns of neural connection and activity spanning numerous neuronal clusters

[1] Parts of this section are adapted from [3], which elaborates on the utility of emulating human cognition on the level of computational theory.

that are highly “fit” in the sense of contributing usefully to system goals. Edelman and his colleagues have run computer simulations showing that Hebbian-like neuronal dynamics, if properly tuned, can give rise to evolution-like dynamics on the neuronal map level (“neuronal group selection”).

Recently Deacon [1] has articulated ways in which, during neurodevelopment, difference computations compete with each other (e.g., to determine which brain regions are responsible for motor control). More generally, he posits a kind of continuous flux as control shifts between competing brain regions, again, based on high-level “cognitive demand” [1, p. 457]. Similarly, Calvin and Bickerton [6, 7] have given plausible neural mechanisms (“Darwin Machines”) for synthesizing short “programs”. These programs are for tasks such as rock throwing and sentence generation, which are represented as coherent firing patterns in the cerebral cortex. A population of such patterns, competing for neurocomputational territory, replicates with variations, under selection pressure to conform to background knowledge and constraints.

In summary, a system is needed that can recombine existing solutions in a non-local synthetic fashion, learning nested and sequential structures, and incorporate background knowledge (e.g. previously learned routines). I propose a particular kind of *program evolution* to satisfy these goals.

1.1. Evolutionary Learning

There is a long history in AI of applying evolution-derived methods to practical problem-solving; the original genetic algorithm [4], initially a theoretical model, has been adapted successfully to a wide variety of applications [8]. The methodology, similar to the Darwin Machines mentioned above, is applied as follows: (1) generate a random population of solutions to a problem; (2) evaluate the solutions in the population using a predefined *scoring function*; (3) select solutions from the population proportionate to their scores, and recombine/mutate them to generate a new population; (4) go to step 2. Holland's paradigm has been adapted from the case of fixed-length strings to the evolution of variable-sized and shaped trees (typically Lisp symbolic expressions), which in principle can represent arbitrary computer programs [9, 10].

Recently, replacements-for/extensions-of the genetic algorithm have been developed (for fixed-length strings) which may be described as *estimation-of-distribution* algorithms (see [11] for an overview). These methods, which outperform genetic algorithms and related techniques across a range of problems, maintain centralized probabilistic models of the population learned with sophisticated datamining techniques. One of the most powerful of these methods is the *Bayesian optimization algorithm* (BOA) [12].

The basic steps of the BOA are: (1) generate a random population of solutions to a problem; (2) evaluate the solutions in the population using a predefined *scoring function*; (3) from the promising solutions in the population, learn a generative model; (4) create new solutions using the model, and merge them into the existing population; (4) go to step 2. The neurological implausibility of this sort of algorithm is readily apparent – yet recall that we are attempting to emulate human cognition on the level of computational theory, not implementation, or even representations and algorithms.

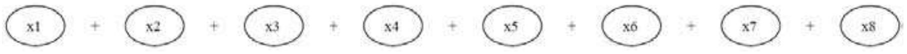


Figure 1: The structure of OneMax, a paradigmatic *separable* optimization problem.

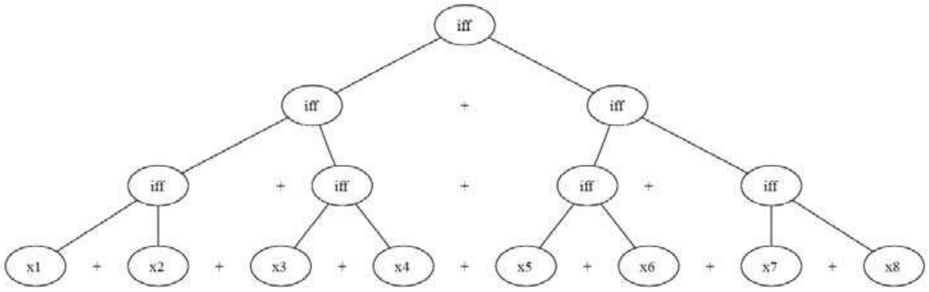


Figure 2: The structure of hierarchical if-and-only-if [16], a paradigmatic nearly decomposable optimization problem.

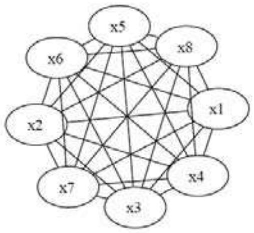


Figure 3: The structure of an intractable optimization problem, such as a uniform random scoring function, where changing the assignment of any variable results in a chaotic change in the overall score.

Fundamentally, the BOA and its ilk (the *competent* adaptive optimization algorithms) differ from classic selectorecombinative search by attempting to dynamically learn a problem decomposition, in terms of the variables that have been pre-specified. The BOA represents this decomposition as a Bayesian network (directed acyclic graph with the variables as nodes, and an edge from x to y indicating that y is probabilistically dependent on x). An extension, the hierarchical Bayesian optimization algorithm (hBOA) [12], uses a Bayesian network with local structure [13] to more accurately represent hierarchical dependency relationships. The BOA and hBOA are scalable and robust to noise across the range of nearly decomposable functions [12, 13]. They are also effective, empirically, on real-world problems with unknown decompositions, which may or may not be effectively representable by the algorithms; robust, high-quality results have been obtained for Ising spin glasses and MaxSAT [14], as well as a real-world telecommunication problem [15].

1.2. Representation-Building

"A representation is a formal system making explicit certain entities or types of information, together with a specification of how the system does this." *David Marr* [2]

In an ideally encoded optimization problem, all prespecified variables would exhibit complete *separability*, and could be optimized independently (Figure 1). Problems with hierarchical dependency structure (Figure 2) cannot be encoded this way, but are still tractable by dynamically learning the problem decomposition (as the BOA and hBOA do). For complex problems with interacting subcomponents, finding an accurate problem decomposition is often tantamount to finding a solution. In an idealized run of a competent optimization algorithm, the problem decomposition evolves along with the set of solutions being considered, with parallel convergence to the correct decomposition and the global solution optima. However, this is certainly contingent on the existence of some compact[2] and reasonably correct decomposition in the space (of decompositions, not solutions) being searched.

Difficulty arises when no such decomposition exists (Figure 3), or when a more effective decomposition exists that cannot be formulated as a probabilistic model over representational parameters. Accordingly, one may extend current approaches via either: (1) a more general modeling language for expressing problem decompositions; or (2) *additional mechanisms* that modify the representations on which modeling operates (introducing additional inductive bias). I focus here on the latter – the former would appear to require qualitatively more computational capacity than will be available in the near future. If one ignores this constraint, such a "universal" approach to general problem-solving is indeed possible [17, 18].

I refer to these additional mechanisms as representation-building because they serve the same purpose as the pre-representational mechanisms employed (typically by humans) in setting up an optimization problem – to present an optimization algorithm with the salient parameters needed to build effective problem decompositions and vary solutions along meaningful dimensions.

A secondary source of human effort in encoding problems to be solved is crafting an effective scoring function. The primary focus of this paper is issues surrounding the representation of solutions, rather than how solutions are scored, which may be more fruitfully addressed, I believe, in the context of integrative systems (cf. [19]).

1.3. Program Learning

An optimization problem may be defined as follows: a solution space $\boldsymbol{S}$ is specified, together with some scoring function on solutions, where "solving the problem" corresponds to discovering a solution in $\boldsymbol{S}$ with a sufficiently high score. Let's define program learning as follows: given a program space $\boldsymbol{P}$, a behavior space $\boldsymbol{B}$, an execution function $\boldsymbol{exec} : \boldsymbol{P} \rightarrow \boldsymbol{B}$, and a scoring function on *behaviors*, "solving the problem" corresponds to discovering a program $\boldsymbol{p}$ in $\boldsymbol{P}$ whose corresponding behavior, $\boldsymbol{exec(p)}$, has a sufficiently high score.

[2] The decomposition must be compact because in practice only a fairly small sampling of solutions may be evaluated (relative to the size of the total space) at a time, and the search mechanism for exploring decomposition-space is greedy and local. This is in also accordance with the general notion of learning corresponding to compression [17].

This extended formalism can of course be entirely vacuous – the behavior space could be identical to the program space, and the execution function simply identity, allowing any optimization problem to be cast as a problem of program learning. The utility of this specification arises when we make interesting assumptions regarding the program and behavior spaces, and the execution and scoring functions (the additional inductive bias mentioned above):

- **Open-endedness** – ***P*** has a natural "program size" measure – programs may be enumerated from smallest to largest, and there is no obvious problem-independent upper bound on program size.
- **Over-representation** – ***exec*** often maps many programs to the same behavior.
- **Compositional hierarchy** – programs themselves have an intrinsic hierarchical organization, and may contain subprograms which are themselves members of ***P*** or some related program space. This provides a natural family of distance measures on programs, in terms of the the number and type of compositions / decompositions needed to transform one program into another (i.e., edit distance).
- **Chaotic Execution** – very similar programs (as conceptualized in the previous item) may have very different behaviors.

Precise mathematical definitions could be given for all of these properties but would provide little insight – it is more instructive to simply note their ubiquity in symbolic representations; human programming languages (LISP, C, etc.), Boolean and real-valued formulae, pattern-matching systems, automata, and many more. The crux of this line of thought is that the combination of these four factors conspires to *scramble* scoring functions – even if the mapping from behaviors to scores is separable or nearly decomposable, the complex[3] program space and chaotic execution function will often quickly lead to intractability as problem size grows. These properties are not superficial inconveniences that can be circumvented by some particularly clever encoding. On the contrary, they are the essential characteristics that give programs the power to compress knowledge and generalize correctly, in contrast to flat, inert representations such as lookup tables (see Baum [20] for a full treatment of this line of argument).

The consequences of this particular kind of complexity, together with the fact that most program spaces of interest are combinatorially very large, might lead one to believe that competent program evolution is impossible. Not so: program learning tasks of interest have a compact structure[4] – they are not "needle in haystack" problems or uncorrelated fitness landscapes, although they can certainly be encoded as such. The most one can definitively state is that algorithm ***foo***, methodology ***bar,*** or representation ***baz*** is unsuitable for expressing and exploiting the regularities that occur across interesting program spaces. Some of these regularities are as follows:

- Simplicity prior – our prior assigns greater probability mass to smaller programs.
- Simplicity preference – given two programs mapping to the same behavior, we prefer the smaller program (this can be seen as a secondary scoring function).

[3] Here "complex" means open-ended, over-representing, and hierarchical.

[4] Otherwise, humans could not write programs significantly more compact than lookup tables.

- Behavioral decomposability – the mapping between behaviors and scores is separable or nearly decomposable. Relatedly, scores are more than scalars – there is a partial ordering corresponding to behavioral dominance, where one behavior dominates another if it exhibits a strict superset of the latter's desideratum, according to the scoring function.[5] This partial order will never contradict the total ordering of scalar scores.
- White box execution – the mechanism of program execution is known a priori, and remains constant across many problems.

How these regularities may be exploited via representation-building, in conjunction with the probabilistic modeling that takes place in a competent optimization algorithm such as the BOA or hBOA, will be discussed in the next section. Another fundamental regularity of great interest for artificial general intelligence, but beyond the scope of this paper, is patterns across related problems that may be solvable with similar programs (e.g., involving common modules).

2. Competent Program Evolution

In this section I present *meta-optimizing semantic evolutionary search* (MOSES), a system for competent program evolution. Based on the viewpoint developed in the previous section, MOSES is designed around the central and unprecedented capability of competent optimization algorithms such as the hBOA, to generate new solutions that simultaneously combine sets of promising assignments from previous solutions according to a dynamically learned problem decomposition. The novel aspects of MOSES described herein are built around this core to exploit the unique properties of program learning problems. This facilitates effective problem decomposition (and thus competent optimization).

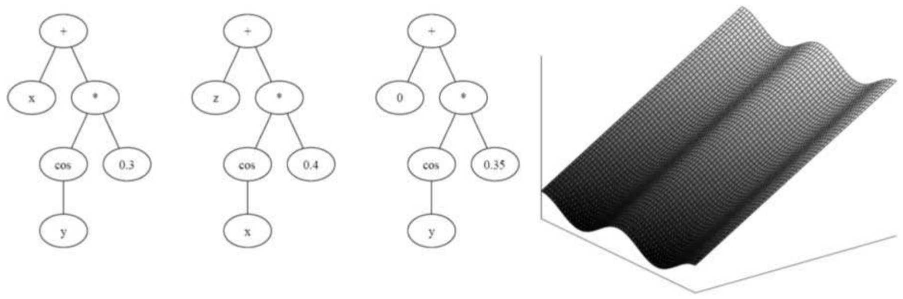


Figure 4: Three simple program trees encoding real-valued expressions, with identical structures and node-by-node semantics (left), and the corresponding behavioral pattern (right); the horizontal axes correspond to variation of arguments (*x*, *y*, and/or *z*), with the vertical axis showing the corresponding variation of output.

[5] For example, in supervised classification one rule dominates another if it correctly classifies all of the items that second rule classifies correctly, as well as some which the second rule gets wrong.

2.1. Statics

The basic goal of MOSES is to exploit the regularities in program spaces outlined in the previous section, most critically *behavioral decomposability* and *white box execution*, to dynamically construct representations that limit and transform the program space being searched into a relevant subspace with a compact problem decomposition. These representations will evolve as the search progresses.

2.1.1. An Example

Let's start with an easy example. What knobs (meaningful parameters to vary) exist for the family of programs depicted in Figure 4 on the left? We can assume, in accordance with the principle of white box execution, that all symbols have their standard mathematical interpretations, and that ***x***, ***y***, and ***z*** are real-valued variables.

In this case, all three programs correspond to variations on the behavior represented graphically on the right in the figure. Based on the principle of behavioral decomposability, good knobs should express plausible evolutionary variation and recombination of features in behavior space, regardless of the nature of the corresponding changes in program space. It's worth repeating once more that this goal cannot be meaningfully addressed on a syntactic level - it requires us to leverage background knowledge of what the symbols in our vocabulary (***cos***, +, ***0.35***, etc.) actually *mean*.

A good set of knobs will also be *orthogonal*. Since we are searching through the space of combinations of knob settings (not a single change at a time, but a set of changes), any knob whose effects are equivalent to another knob or combination of knobs is undesirable.[6] Correspondingly, our set of knobs should *span* all of the given programs (i.e., be able to represent them as various knob settings).

A small *basis* for these programs could be the 3-dimensional parameter space, $x_1 \in \{x, z, 0\}$ (left argument of the root node), $x_2 \in \{y, x\}$ (argument of ***cos***), and $x_3 \in [0.3, 0.4]$ (multiplier for the ***cos***-expression). However, this is a very limiting view, and overly tied to the particulars of how these three programs happen to be encoded. Considering the space *behaviorally* (right of Figure 4), a number of additional knobs can be imagined which might be turned in meaningful ways, such as:

1. numerical constants modifying the phase/frequency of the cosine expression,
2. considering some weighted average of x and y instead of one or the other,
3. multiplying the entire expression by a constant,
4. adjusting the relative weightings of the two arguments to +.

2.1.2. Syntax and Semantics

This kind of representation-building calls for a correspondence between syntactic and semantic variation. The properties of program spaces that make this difficult are over-representation and chaotic execution, which lead to *non-orthogonality*, *oversampling of distant behaviors*, and *undersampling of nearby behaviors*, all of which can directly impede effective program evolution.

[6] First because this will increase the number of samples needed to effectively model the structure of knob-space, and second because this modeling will typically be quadratic with the number of knobs, at least for the BOA or hBOA [12, 13].

Non-orthogonality is caused by over-representation. For example, based on the properties of commutativity and associativity, $a_1 + a_2 + ... + a_n$ may be expressed in exponentially many different ways, if + is treated as a non-commutative and non-associative binary operator. Similarly, operations such as addition of zero and multiplication by one have no effect, the successive addition of two constants is equivalent to the addition of their sum, etc. These effects are not quirks of real-valued expressions; similar redundancies appear in Boolean formulae (***x AND x*** $\leftrightarrow$ ***x***), list manipulation (***cdr(cons(x, L))*** $\leftrightarrow$ ***L***), and conditionals (***if x then y else z*** $\leftrightarrow$ ***if NOT x then z else y***).

Without the ability to exploit these identities, we are forced to work in a greatly expanded space which represents equivalent expression in many different ways, and will therefore be very far from orthogonality. Completely eliminating redundancy is infeasible, and typically at least NP-hard (in the domain of Boolean formulae it is reducible to the satisfiability problem, for instance), but one can go quite far with a heuristic approach.

Oversampling of distant behaviors is caused directly by chaotic execution, as well as a somewhat subtle effect of over-representation, which can lead to simpler programs being heavily oversampled. Simplicity is defined relative to a given program space in terms of minimal length, the number of symbols in the shortest program that produces the same behavior.

Undersampling of nearby behaviors is the flip side of the oversampling of distant behaviors. As we have seen, syntactically diverse programs can have the same behavior; this can be attributed to redundancy, as well as non-redundant programs that simply compute the same result by different means. For example, ***3*x*** can also be computed as ***x+x+x***; the first version uses less symbols, but neither contains any obvious “bloat” such as addition of zero or multiplication by one. Note however that the nearby behavior of ***3.1*x***, is syntactically close to the former, and relatively far from the latter. The converse is the case for the behavior of ***2*x+y***. In a sense, these two expressions can be said to exemplify differing organizational principles, or points of view, on the underlying function.

Differing organizational principles lead to different biases in sampling nearby behaviors. A superior organizational principle (one leading to higher-scoring syntactically nearby programs for a particular problem) might be considered a *metaptation* (adaptation at the second tier), in the terminology of King [21]. Since equivalent programs organized according to different principles will have identical scores, some methodology beyond selection for high scores must be employed to search for good organizational principles. Thus, the resolution of undersampling of nearby behaviors revolves around the management of *neutrality* in search, a complex topic beyond the scope of this paper.

These three properties of program spaces greatly affect the performance of evolutionary methods based solely on syntactic variation and recombination operators, such as local search or genetic programming. In fact, when quantified in terms of various fitness-distance correlation measures, they can be effective predictors of algorithm performance, although they are of course not the whole story [22]. A semantic search procedure will address these concerns in terms of the underlying behavioral effects of and interactions between a language's basic operators; the general scheme for doing so in MOSES is the topic of the next subsection.

2.2. Neighborhoods and Normal Forms

The procedure MOSES uses to construct a set of knobs for a given program (or family of structurally related programs) is based on three conceptual steps: *reduction to normal form*, *neighborhood enumeration*, and *neighborhood reduction*.

Reduction to normal form - in this step, redundancy is heuristically eliminated by reducing programs to a *normal form*. Typically, this will be via the iterative application of a series of local rewrite rules (e.g., $\forall x, x+0 \rightarrow x$), until the target program no longer changes. Note that the well-known conjunctive and disjunctive normal forms for Boolean formulae are generally unsuitable for this purpose; they destroy the hierarchical structure of formulae, and dramatically limit the range of behaviors (in this case Boolean functions) that can be expressed compactly. Rather, *hierarchical normal forms* for programs are required.

Neighborhood enumeration - in this step, a set of possible atomic *perturbations* is generated for all programs under consideration (the overall perturbation set will be the union of these). The goal is to heuristically generate new programs that correspond to behaviorally nearby variations on the source program, in such a way that arbitrary sets of perturbations may be *composed* combinatorially to generate novel valid programs.

Neighborhood reduction - in this step, redundant perturbations are heuristically culled to reach a more orthogonal set. A straightforward way to do this is to exploit the reduction to normal form outlined above; if multiple knobs lead to the same normal forms program, only one of them is actually needed. Additionally, note that the number of symbols in the normal form of a program can be used as a heuristic approximation for its minimal length - if the reduction to normal form of the program resulting from twiddling some knob significantly decreases its size, it can be assumed to be a source of oversampling, and hence eliminated from consideration. A slightly smaller program is typically a meaningful change to make, but a large reduction in complexity will rarely be useful (and if so, can be accomplished through a combination of knobs that individually produce small changes).

At the end of this process, we will be left with a set of knobs defining a subspace of programs centered around a particular region in program space and heuristically centered around the corresponding region in behavior space as well. This is part of the *meta* aspect of MOSES, which seeks not to evaluate variations on existing programs itself, but to construct parameterized program subspaces (representations) containing meaningful variations, guided by background knowledge. These representations are used as search spaces within which an optimization algorithm can be applied.

2.3. Dynamics

As described above, the representation-building component of MOSES constructs a parameterized representation of a particular *region* of program space, centered around a single program of family of closely related programs. This is consonant with the line of thought developed above, that a representation constructed across an arbitrary region of program space (e.g., all programs containing less than ***n*** symbols), or spanning an arbitrary collection of unrelated programs, is unlikely to produce a meaningful parameterization (i.e., one leading to a compact problem decomposition).

A sample of programs within a region derived from representation-building together with the corresponding set of knobs will be referred to herein as a *deme*;[7] a set of demes (together spanning an arbitrary area within program space in a patchwork fashion) will be referred to as a *metapopulation.* [8] MOSES operates on a metapopulation, adaptively creating, removing, and allocating optimization effort to various demes. Deme management is the second fundamental *meta* aspect of MOSES, after (and above) representation-building; it essentially corresponds to the problem of effectively allocating computational resources to competing regions, and hence to competing programmatic organizational- representational schemes.

2.4. Algorithmic Sketch

The salient aspects of programs and program learning lead to requirements for competent program evolution that can be addressed via a representation-building process such as the one shown above, combined with effective deme management. The following sketch of MOSES presents a simple control flow that dynamically integrates these processes into an overall program evolution procedure:

1. Construct an initial set of knobs based on some prior (e.g., based on an empty program) and use it to generate an initial random sampling of programs. Add this deme to the metapopulation.

2. Select a deme from the metapopulation and update its sample, as follows:
   a) Select some promising programs from the deme's existing sample to use for modeling, according to the scoring function.
   b) Considering the promising programs as collections of knob settings, generate new collections of knob settings by applying some (competent) optimization algorithm.
   c) Convert the new collections of knob settings into their corresponding programs, reduce the programs to normal form, evaluate their scores, and integrate them into the deme's sample, replacing less promising programs.

3. For each new program that meets the criterion for creating a new deme, if any:
   a) Construct a new set of knobs (via representation-building) to define a region centered around the program (the deme's *exemplar)*, and use it to generate a *new* random sampling of programs, producing a new deme.
   b) Integrate the new deme into the metapopulation, possibly displacing less promising demes.

4. Repeat from step 2.

[7] A term borrowed from biology, referring to a somewhat isolated local population of a species.

[8] Another term borrowed from biology, referring to a group of somewhat separate populations (the demes) that nonetheless interact.

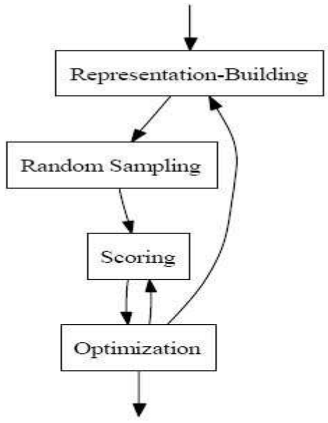


Figure 5: The top-level architectural components of MOSES, with directed edges indicating the flow of information and program control.

```
→###
   #
   #                      ###
   #                     #    #
   #                     #    #
   #### #####         ##
            #                 #
            #        #
            #        #
            #        #        #
                     #
            #
            #                 #
            #        #
            #        #     ###
                  #     #
```

Figure 6: The top half of the Santa Fe trail. The ant's starting position and orientation is denoted by the →, and pieces of food by #s. The complete trail may be found in [10, p. 55].

The criterion for creating a new deme is *behavioral non-dominance* (programs which are not dominated by the exemplars of any existing demes are used as exemplars to create new demes), which can be defined in a domain-specific fashion. As a default, the scoring function may be used to induce dominance, in which case the set of exemplar programs for demes corresponds to the set of top-scoring programs.

2.5. Architecture

The preceding algorithmic sketch of MOSES leads to the top-level architecture depicted in Figure 5. Of the four top-level components, only the scoring function is problem-specific. The representation-building process is domain-specific, while the random sampling methodology and optimization algorithm are domain-general. There is of course the possibility of improving performance by incorporating domain and/or problem-specific bias into random sampling and optimization as well.

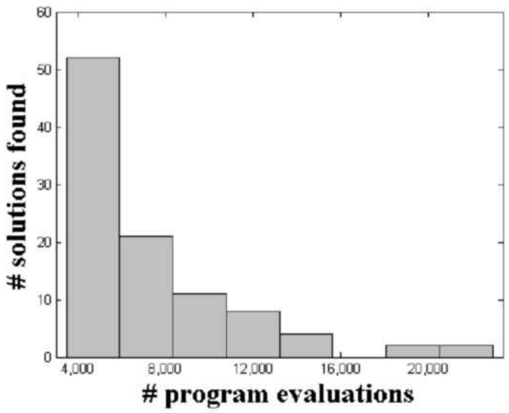
and third rules are fully general simplification rules based on the semantics of *if-then-else* statements and associative functions (such as *progn*), respectively.

These rules allow us to naturally parameterize a knob space corresponding to a given program (note that the arguments to the *progn* and *if-food-ahead* functions will be recursively reduced and parameterized according to the same procedure). Rotations will correspond to knobs with four possibilities (*left, right, reversal, no rotation)*. Movement commands will correspond to knobs with two possibilities (*move, no movement)*. There is also the possibility of introducing a new command in between, before, or after, existing commands. Some convention (a "canonical form") for our space is needed to determine how the knobs for new commands will be introduced. A representation consists of a rotation knob, followed by a conditional knob, followed by a movement knob, followed by a rotation knob, etc.[10]

The structure of the space (how large and what shape) and default knob values will be determined by the "exemplar" program used to construct it. The default values are used to bias the initial sampling to focus around the exemplar: all of the *n* neighbors of the exemplar are first added to the sample, followed by a random selection of *n* programs at a distance of two from the exemplar, *n* programs at a distance of three, etc., until the entire sample is filled. Note that the hBOA can of course effectively recombine this sample to generate novel programs at any distance from the exemplar. The empty program *progn* (which can be used as the initial exemplar for MOSES), for example, leads to the following prototype subspace:

```
progn(
        rotate? [default no rotation],
        if-food-ahead(
                progn(
                        rotate? [default no rotation],
                        move? [default no movement]),
                progn(
                        rotate? [default no rotation],
                        move? [default no movement])),
        move? [default no movement])
```

[10] That there be some fixed ordering on the knobs is important, so that two rotation knobs are not placed next to each other (as this would introduce redundancy). Based on some preliminary test, the precise ordering chosen (rotation, conditional, movement) does not appear to be critical.



Technique	Computational Effort
Genetic Programming [10]	450,000 evaluations
Evolutionary Programming [24]	136,000 evaluations
MOSES	23,000 evaluations

Figure 7: On the left, histogram of the number of global optima found after a given number of program evaluations for 100 runs of MOSES on the artificial ant problem (each run is counted once for the first global optimum reached). On the right, computational effort required to find an optimal solution for various techniques with p=.99 (for MOSES p=1, since an optimal solution was found in all runs).

There are six parameters here, three which are quaternary, and three which are binary. So the program *progn(left,if-food-ahead(move, left))* would be encoded in the space as [*left, no rotation, move, left, no movement, no movement*], with knobs ordered according to a pre-order left-to-right traversal of the program's parse tree (this is merely for exposition; the ordering of the parameters has no effect on MOSES). For an exemplar program already containing an *if-food-ahead* statement, nested conditionals would be considered.

A space with six parameters in it is small enough that MOSES can reliably find the optimum (the program *progn(right, if-food-ahead(progn(),left), move)*), with a very small population. After no further improvements have been made in the search for a specified number of generations (calculated based on the size of the space based on a model derived from [23] that is general to the hBOA, and not at all tuned for the artificial ant problem), a new representation is constructed centered around this program.[11] Additional knobs are introduced "in between" all existing ones (e.g., an optional move in between the first rotation and the first conditional), and possible nested conditionals are considered (a nested conditional occurring in a sequence *after* some other action has been taken is not redundant). The resulting space has 39 knobs, still quite tractable for hBOA, which typically finds a global optimum within a few generations. If the optimum were not to be found, MOSES would construct a new (possibly larger or smaller) representation, centered around the best program that *was* found, and the process would repeat.

The artificial ant problem is well-studied, with published benchmark results available for genetic programming [10] as well as evolutionary programming based solely on mutation [24] (i.e., a form of population-based stochastic hill climbing).

[11] MOSES reduces the exemplar program to normal form before constructing the representation; in this particular case however, no transformations are needed. Similarly, in general neighborhood reduction would be used to eliminate any extraneous knobs (based on domain-specific heuristics). For the ant domain however no such reductions are necessary.

Furthermore, an extensive analysis of the search space has been carried out by Langdon and Poli [25], with the authors concluding:

- The problem is “deceptive at all levels”, meaning that the partial solutions that must be recombined to solve the problem to optimality have lower average fitness than the partial solutions that lead to inferior local optima.
- The search space contains many symmetries (e.g., between left and right rotations),
- There is an unusually high density of global optima in the space (relative to other common test problems); even though current evolutionary methods can solve the problem, they are not significantly more effective (in terms of the number of program evaluations require) than random sample.
- “If real program spaces have the above characteristics (we expect them to do so but be still worse) then it is important to be able to demonstrate scalable techniques on such problem spaces”.

A review of scalability results for MOSES across a range of problems is beyond the scope of this paper (see [26]), but results for the artificial ant problem may be given briefly to indicate the magnitude of improvement that may be experienced. Koza [10] reports on a set of 148 runs of genetic programming with a population size of 500 which had a 16% success rate after 51 generations when the runs were terminated (a total of 25,500 program evaluations per run). The minimal “computational effort” needed to achieve success with 99% probability was attained by processing through generation 14 was 450,000 (based on parallel independent runs). Chellapilla [24] reports 47 out of 50 successful runs with a minimal computational effort (again, for success with 99% probability) of 136,000 for his stochastic hill climbing method.

One hundred runs of MOSES were executed. Beyond the domain knowledge embodied in the reduction and knob construction procedure, the only parameter that needed to be set was the population scaling factor, which was set to 30 (MOSES automatically adjusts to generate a larger population as the size of the representation grows, with the base case determined by this factor). Based on these “factory” settings, MOSES found optimal solutions on every run out of 100 trials, within a maximum of 23,000 program evaluations (the computational effort figure corresponding to 100% success). The average number of program evaluation required was 6952, with 95% confidence intervals of ±856 evaluations.

Why does MOSES outperform other techniques? One factor to consider first is that the language programs are evolved in is slightly more expressive than that used for the other techniques; specifically, a *progn* is allowed to have no children (if all of its possible children are “turned off”), leading to the possibility of *if-food-ahead* statements which do nothing if food is present (or not present). Indeed, many of the smallest solutions found by MOSES exploit this feature. This can be tested by inserting a “do nothing” operation into the terminal set for genetic programming (for example). Indeed, this reduces the computational effort to 272,000; an interesting effect, but still over an order of magnitude short of the results obtained with MOSES (the success rate after 50 generations is still only 20%).

Another possibility is that the reductions in the search space via simplification of programs alone are responsible. However, the results past attempts at introducing program simplification into genetic programming systems [27, 28] have been mixed; although the system may be sped up (because programs are smaller), there have been no dramatic improvement in results noted. To be fair, these results have been primarily focused on the symbolic regression domain; I am not aware of any results for the artificial ant problem.

The final contributor to consider is the sampling mechanism (knowledge-driven knob-creation followed by probabilistic model-building). We can test to what extent model-building contributes to the bottom line by simply disabling it and assuming probabilistic independence between all knobs. The result here is of interest because model-building can be quite expensive ($O(n^2N)$ per generation, where n is the problem size and N is the population size[12]). In 50 independent runs of MOSES without model-building, a global optimum was still discovered in all runs. However, the variance in the number of evaluations required was much higher (in two cases over 100,000 evaluations were needed). The new average was 26,355 evaluations to reach an optimum (about 3.5 times more than required with model-building). The contribution of model-building to the performance of MOSES is expected to be even greater for more difficult problems.

Applying MOSES without model-building (i.e., a model assuming no interactions between variables) is a way to test the combination of representation-building with an approach resembling the probabilistic incremental program learning (PIPE) [29] algorithm, which learns programs based on a probabilistic model without any interactions. PIPE has no been shown to provide results competitive with genetic programming on a number of problems (regression, agent control, etc.).

It is additionally possible to look inside the models that the hBOA constructs (based on the empirical statistics of successful programs) to see what sorts of linkages between knobs are being learned.[13] For the 6-knob model given above for instance an analysis the linkages learned shows that the three most common pairwise dependencies uncovered, occurring in over 90% of the models across 100 runs, are between the rotation knobs. No other individual dependencies occurred in more than 32% of the models. This preliminary finding is quite significant given Landgon and Poli's findings on symmetry, and their observation [25] that "[t]hese symmetries lead to essentially the same solutions appearing to be the opposite of each other. E.g. either a pair of Right or pair of Left terminals at a particular location may be important."

In summary, all of the components of MOSES appear to mesh together to provide superior performance, although further experimentation and analysis across a range of problems is clearly needed.

2.7. Discussion

The overall MOSES design as described herein is unique. However, it is instructive at this point to compare its two primary facets (representation-building and deme management) to related work in evolutionary computation.

Rosca's *adaptive representation* architecture [30] is an approach to program evolution which also alternates between separate representation-building and optimization stages. It is based on Koza's genetic programming [10], and modifies the representation based on a *syntactic* analysis driven by the scoring function, as well as a modularity bias. The representation-building that takes place consists of introducing new compound operators, and hence modifying the implicit distance function in tree-

[12] The fact that reduction to normal tends to reduce the problem size is another synergy between it and the application of probabilistic model-building.

[13] There is in fact even more information available in the hBOA models concerning hierarchy and direction of dependence, but this is difficult to analyze.

space. This modification is uniform, in the sense that the new operators can be placed in any context, without regard for semantics.

In contrast to Rosca's work and other approaches to representation-building such as Koza's *automatically defined functions* [31], MOSES explicitly addresses on the underlying (semantic) structure of program space independently of the search for any kind of modularity or problem decomposition. This preliminary stage critically changes neighborhood structures (syntactic similarity) and other aggregate properties of programs.

Regarding deme management, the embedding of an evolutionary algorithm within a superordinate procedure maintaining a metapopulation is most commonly associated with "island model" architectures [8]. One of the motivations articulated for using island models has been to allow distinct islands to (usually implicitly) explore different regions of the search space, as MOSES does explicitly. MOSES can thus be seen as a very particular kind of island model architecture, where programs never migrate between islands (demes), and islands are created and destroyed dynamically as the search progresses.

In MOSES, optimization does not operate directly on program space, but rather on a subspace defined by the representation-building process. This subspace may be considered as being defined by a sort of template assigning values to some of the underlying dimensions (e.g., it restricts the size and shape of any resulting trees). The messy genetic algorithm [32], an early competent optimization algorithm, uses a similar mechanism - a common "competitive template" is used to evaluate candidate solutions to the optimization problem which are themselves underspecified. Search consequently centers on the template(s), much as search in MOSES centers on the programs used to create new demes (and thereby new representations). The issue of deme management can thus be seen as analogous to the issue of template selection in the messy genetic algorithm.

3. Summary and Conclusions

Competent evolutionary optimization algorithms are a pivotal development, allowing encoded problems with compact decompositions to be tractably solved according to normative principles. We are still faced with the problem of *representation-building* – casting a problem in terms of knobs that can be twiddled to solve it. Hopefully, the chosen encoding will allow for a compact problem decomposition. Program learning problems in particular rarely possess compact decompositions, due to particular features generally present in program spaces (and in the mapping between programs and behaviors). This often leads to intractable problem formulations, even if the mapping between behaviors and scores has an intrinsic separable or nearly decomposable structure. As a consequence, practitioners must often resort to manually carrying out the analogue of representation-building, on a problem-specific basis.

Working under the thesis that *the properties of programs and program spaces can be leveraged as inductive bias to reduce the burden of manual representation-building, leading to competent program evolution*, I have developed the MOSES system. Experimental results and theoretical analyses have been carried out to substantiate many of the claims made in this paper regarding the properties of program spaces and how to transform them via representation-building, which will be presented in [26], along with results and analyses of MOSES itself; order-of-magnitude reductions in the number of scoring function evaluations needed to find an optimal solution have already been achieved relative to genetic programming [10], on a number of benchmarks, in addition to the ant results presented herein. Future work will focus on advanced problem domains (higher-order functions, list manipulation, recursion, etc.), more adaptive representation-building for better scalability, exploiting additional inductive bias in the behavioral space, and, most importantly in the long-term, integration with other AI components.

Acknowledgements

Thanks to Ben Goertzel for many suggestions and discussions which have been instrumental in developing the ideas presented in this paper. Thanks to the anonymous reviewer for constructive suggestions that have hopefully made this a more cogent and comprehensible paper than it would have been otherwise.

References

[1] T. Deacon. *The Symbolic Species*. Norton, 1997.
[2] D. Marr. *Vision: a computational investigation into the human representation and processing of visual information*. W. H. Freeman, 1982.
[3] M. Looks and B. Goertzel. Mixing cognitive science concepts with computer science algorithms and data structures: An integrative approach to strong AI. In *AAAI Spring Symposium Series*, 2006.
[4] J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.
[5] G. M. Edelman. *Neural Darwinism: The Theory of Neuronal Group Selection*. Basic Books, 1988.
[6] W. H. Calvin. The brain as a Darwin machine. *Nature*, 1987.
[7] W. H. Calvin and D. Bickerton. *Lingua ex Machina*. MIT Press, 2000.
[8] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1989.
[9] N. Cramer. A representation for the adaptive generation of simple sequential programs. In *International Conference on Genetic Algorithms and their Applications*, 1985.
[10] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.
[11] M. Pelikan, D. E. Goldberg, and F. G. Lobo. A survey of optimization by building and using probabilistic models. *Computational Optimization and Applications*, 2002.
[12] M. Pelikan, D. E. Goldberg, and E. Cantú-Paz. Linkage problem, distribution estimation, and Bayesian networks. *Evolutionary Computation*, 2002.
[13] M. Pelikan and D. E. Goldberg. A hierarchy machine: Learning to optimize from nature and humans. *Complexity*, 2003.
[14] M. Pelikan and D. E. Goldberg. Hierarchical BOA solves Ising spin glasses and MAXSAT. Technical report, University of Illinois at Urbana-Champaign, 2003.
[15] F. Rothlaud, D. E. Goldberg, and A. Heinzl. Bad codings and the utility of well-designed genetic algorithms. Technical report, University of Illinois at Urbana-Champaign, 2000.
[16] R. A. Watson, G. S. Hornby, and J. B. Pollack. Modeling building-block interdependency. In Parallel Problem Solving from Nature, 1998.
[17] M. Hutter. Universal algorithmic intelligence: A mathematical top-down approach. In B. Goertzel and C. Pennachin, editors, *Artificial General Intelligence*. Springer-Verlag, 2005.
[18] J. Schmidhuber. Gödel machines: Fully self-referential optimal universal self-improvers. In B. Goertzel and C. Pennachin, editors, *Artificial General Intelligence*. Springer-Verlag, 2005.
[19] M. Looks, B. Goertzel, and C. Pennachin. Novamente: An integrative architecture for artificial general intelligence. In *AAAI Fall Symposium Series*, 2004.
[20] E. B. Baum. *What is Thought?* MIT Press, 2004.
[21] D.G. King. Metaptation: the product of selection at the second tier. *Evolutionary Theory*, 1985.
[22] M. Tomassini, L. Vanneschi, P. Collard, and M. Clergue. A study of fitness distance correlation as a difficulty measure in genetic programming. *Evolutionary Computation*, 2005.

[23] M. Pelikan and T. Lin. Parameter-less hierarchical BOA. In Genetic and Evolutionary Computation Conference, 2004.
[24] K. Chellapilla. Evolving computer programs without subtree crossover. *IEEE Transactions on Evolutionary Computation,* 1997.
[25] W. B. Langdon and R. Poli. Why ants are hard. In Genetic Programming, 1998.
[26] M. Looks. *Competent Program Evolution*. PhD thesis, Washington University in St. Louis, 2006 (forthcoming).
[27] A. Ekárt. Shorter Fitness Preserving Genetic Programming. In Artificial Evolution, 2000.
[28] P. Wong and M. Zhang. Algebraic Simplification of GP Programs During Evolution. In Genetic and Evolutionary Computation Conference, 2006.
[29] R. P. Salustowicz and J. Schmidhuber. Probabilistic incremental program evolution. *Evolutionary Computation*, 1997.
[30] J. Rosca. *Hierarchical Learning with Procedural Abstraction Mechanisms*. PhD thesis, University of Rochester, 1997.
[31] J. R. Koza. *Genetic Programming II: Automatic Discovery of Reusable Programs*. MIT Press, 1994.
[32] D. E. Goldberg, B. Korb, and K. Deb. Messy genetic algorithms: Motivation, analysis, and first results. *Complex Systems*, 1989.

B. Goertzel and P. Wang (Eds.)
IOS Press, 2007
© 2007 The authors and IOS Press. All rights reserved.

Artificial Brains. An Inexpensive Method for Accelerating the Evolution of Neural Network Modules for Building Artificial Brains

Hugo de GARIS[a], Liu RUI[a], Huang DI[b], Hu JING[c]

[a] *Brain Builder Group, Key State Laboratory of Software Engineering, Wuhan University, Wuhan, Hubei Province, CHINA*
[b] *Evolvable Hardware Group, School of Computer Science, China University of Geosciences, Wuhan, Hubei Province, CHINA*
[c] *Computer Science Department, Utah State University, Logan, Utah, USA*

profhugodegaris@yahoo.com, pnicholas@vip.sina.com, aaron192032@gmail.com, jinghu78@gmail.com

Abstract. This chapter shows how a "Celoxica" electronic board (containing a Xilinx Virtex II FPGA chip) can be used to accelerate the evolution of neural network modules that are to be evolved quickly enough, so that building artificial brains that consist of 10,000s of interconnected modules, can be made practical. We hope that this work will prove to be an important stepping stone towards making the new field of brain building both practical and cheap enough for many research groups to start building their own artificial brains.

Introduction

The primary research goal of the first author is to build *artificial brains* [1]. An artificial brain is defined to be a collection of interconnected neural net modules (10,000–50,000 of them), each of which is evolved quickly in special electronic hardware, downloaded into a PC, and interconnected according to the designs of human BAs (brain architects). The neural signaling of the artificial brain (A-Brain) is performed by the PC in real time (defined to be 25Hz per neuron). Such artificial brains can be used for many purposes, e.g. controlling the behaviors of autonomous robots.

There is at least one major problem with the above vision, and that is the slow evolution time of individual neural network modules. Typically, it can take many hours to even half a day to evolve a neural net module on a PC. Obviously, evolving several tens of thousands of such modules using only a PC to build an artificial brain will not be practical. Before such A-Brains can be built with this approach, it will be necessary to find ways to accelerate the evolution of such a large number of neural net (NN) modules. At the present time, the authors are pursuing the following approach. We perform the NN module evolution in hardware, to achieve a speedup factor (relative to ordinary PC evolution speeds) of about 10-50 (the topic of this chapter).

We use a Celoxica company's FPGA electronic board (containing a 3 megagate FPGA, i.e. Xilinx's Virtex II chip) to accelerate the evolution of neural network modules. One of the aims of this chapter is to report on the measurement of the speedup factor when using this Celoxica board to evolve neural network modules, compared to using a PC, as performed on the same evolution task.

The first experiment reported on in this chapter involved the evolution of a fairly simple neural net. This task was chosen to serve as the basis for the above comparison and is described in detail in section 2. The task itself was evolved 10 times and an average evolution time calculated. The evolutionary task used two different approaches and technologies, namely a) a standard genetic algorithm on a PC, and b) using the Celoxica board, and the high level language "Handel-C" [2] to program the evolution. Once the two execution time measurements were obtained, the speedup factor could be calculated. Final results are shown in section 7.

1. The Evolution of Neural Network Modules

This section gives a brief description of the approach we use normally to evolve our neural network (NN) modules, to be used as components in building artificial brains. We use a particular neural net model called"GenNet" [3]. A GenNet neural network consists of N (typically N = 12-20) fully connected artificial neurons. Each of the N^2 connections has a weight, represented as a signed binary fraction, with p (typically p = 6-10) bits per weight. The bit string chromosome used to evolve the N^2 weights will have a length of N^2*(p+1) bits. Each neuron "j" receives input signals from the N neurons (i.e. including a signal from itself). Each input signal Sij is multiplied by the corresponding connection weight Wij and summed. To this sum is added an external signal value Ej. This final sum is called the activation signal Aj to the neuron "j".

$$Aj = \sum_{i=1}^{N} WijSij + Ej$$

This activation value is fed into a sigmoid function g that acts as a "squashing" function, limiting the output value Sj to have a maximum absolute value of 1.0

$$Sj = g(Aj) = \frac{Aj}{|Aj| + 1.0}$$

2. The Evolutionary Task

In order to compare the evolution times for the two different approaches (i.e. using an ordinary GA on a PC, and using the Celoxica board), the same fairly simple neural net task was evolved, namely to output a constant signal value of 0.8 over 100 clock ticks.

A single neuron of the network was randomly chosen to be the output neuron for the whole network. Its output signal S(t) was compared to a target (desired) signal value for each of T (=100) clock ticks. The fitness function was defined as follows.

$$f = \frac{1}{\sum_{t=1}^{100} (T(t) - S(t))^2}$$

In our evolutionary task, the target value T(t) is constant, i.e. T(t) = 0.8

In order to compare the evolution times of the above task, a concrete “cutoff” fitness value was used. This was found empirically, by evolving the standard GA version to what we felt was a reasonably “saturated” fitness value. This fitness value was then used in the Celoxica evolution experiments. Once the evolution, as specified by the Handel-C program executed on the Celoxica board reached the same fitness value, its evolutiontime (in seconds) was recorded. The task was evolved 10 times and the average value calculated.

3. Our Evolutionary Approach

The approach we pursue in this chapter to accelerating the evolution time of neural network modules is to perform the evolution in special hardware, e.g. using a Celoxica board [4]. In this approach, a high level language, called Handel-C [Handel-C 2006] is used, whose code is hardware compiled (silicon compiled) into the FPGA (Xilinx’s Virtex II chip). We continue to work on this approach. In this first experiment, we have achieved an 8-fold speedup up compared with a standard GA algorithm on an ordinary PC. By using pipelining techniques and a larger FPGA, we hope to achieve a speedup factor of up to 50 times for the same task. (As will be seen in our more recent experiments, by evolving a smaller neural net, we were actually able to achieve this 50-fold speedup. See section 8).

In order to calculate the speedup factor of the Celoxica board relative to an ordinary genetic algorithm on a PC, the same evolutionary task specified in section 2 was used. The next section gives a brief description of the ordinary genetic algorithm. Section 5 described briefly the characteristics of the Celoxica board. Section 6 introduces the high level language Handel-C, used to program the Celoxica board. Section 7 presents the result of the comparison. Section 8 presents the results of our more recent evolutionary experiments on smaller neural nets. Section 9 presents ideas for future work, and section 10 summarizes.

4. Description of the Standard Genetic Algorithm

The standard GA (Genetic Algorithm) used to help calculate the speed up factor consisted of the following steps.

a) Randomly generate 100 bit string chromosomes of length $N^2*(p+1)$. Over the years we have used values, N = 16, p = 8, so our chromosomes (bit strings) were 2304 bits long.

b) Decode the chromosome into the N^2 signed binary fraction weights, and build the neural network for each chromosome.

c) Perform the fitness measurements for the task concerned. For details, see the next section.

d) Rank the fitnesses from best to worst.

e) Throw out the inferior half of the chromosomes. Replace with the superior half.

f) Mutate all the chromosomes except the top one. No crossover was performed in these experiments.

g) Go to b), until the evolution saturates at the target (desired) fitness value (of the elite chromosome).

5. The Celoxica Board

The aims of lowering the price of high-speed evolution, and achieving higher performance in evolving hardware led us to choose to use FPGAs (Field Programmable Gate Arrays). FPGAs are specially made digital semiconductor circuits that are often used for prototyping. The several million logic gates in modern FPGAs (e.g. Xilinx's Virtex II chip) make it is possible to have multiple copies of the same electronic sub circuit running simultaneously on different areas of the FPGA. This parallelism is very useful for a genetic algorithm. It allows the program to process the most time costly weight calculations in parallel, and this can speed up the overall evolution by a factor of tens to hundred of times.

We chose the Celoxica FPGA board for our project. "Celoxica" is the name of a UK company [4]. Our Celoxica board (an RC203) cost about $1500. With such a board, a design engineer is able to *program* electrical connections on site for a specific application, without paying thousands of dollars to have the chip manufactured in mass quantities. [4].

The RC Series Platforms of Celoxica are complete solutions for FPGA design and ASIC/SoC (system on a chip) verification and prototyping. The boards combine very high-density FPGA devices with soft-core & hard-core processors and an extensive array of peripherals. They provide easy access to programmable SoC's from the Electronic System Level (ESL) and consistently deliver fast and efficient access to very high-density reconfigurable silicon.

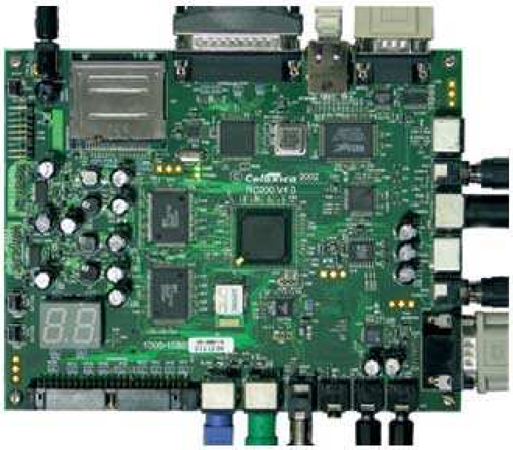


Fig.1. The Celoxica Board with central FPGA

We are currently using an RC203 FPGA board in our experiment. It's a desktop platform for the evaluation and development of high performance applications. The main FPGA chip is a Xilinx Virtex II that can be configured without using an HDL (Hardware Description Language). Instead it uses a much easier high-level "C-like" language called "Handel-C" (after Handel the composer). This language is very similar to ordinary C (i.e. with approximately an 80% overlap between the two languages), with a few extra features, particularly those involved with specifying which functions ought to be executed in *parallel*.

With several million logic gates in the Virtex FPGA chip, it is possible to have multiple copies of the same electronic (sub) circuit running simultaneously. This parallelism allows a genetic algorithm for example to run much faster than on a PC. A Celoxica board attaches to a PC, with two-way communication, so that instructions to the board come from the PC, and results coming from the board can be displayed on the PC.

At the time of writing (October 2006), experiments are continuing to determine how much faster the evolution of a neural net module on the Celoxica board can be, compared to using a software approach in a PC. The value of this speedup factor is *critical* to this whole approach. Colleagues in the ECE (Electronic and Computer Engineering) departments at our respective universities, who are experienced in using Celoxica boards, estimate that the speed up factors will range between 10s and 100s of times. If these speedup factors can be achieved, it then becomes practical to evolve large numbers of neural net modules in a reasonable time, and connect them together to build artificial brains inside a PC.

6. The High Level Language "Handel-C"

Handel-C is Celoxica's high-level language used to configure the Virtex-II FPGA chip on the Celoxica board. This language is mostly ordinary C, with a few parallel programming features, e.g. the "par" statement, that takes the following general form:

```
par
{
   statement A;
   statement B;
   statement C;
}
```

The above Handel-C statement will cause all the statements (i.e. A, B, C) in the par block to be executed at once, by having each statement be configured, with its own separate set of logic gates in the Virtex-II chip on the Celoxica board. By using multiple copies of neural net modules, i.e. multiple electronic copies, each with its own area of programmable silicon, evolving in parallel, it is possible to speed up the overall evolution time, compared to ordinary software based PC evolution. In the following, we introduce the features of the Handle-C language.

Handel-C Programs

Handel-C uses much of the syntax of conventional C with the addition of inherent parallelism. You can write sequential programs in Handel-C, but to gain maximum benefit in performance from the target hardware you must use its parallel constructs.

Handel-C provides constructs to control the flow of a program. For example, code can be executed conditionally depending on the value of some expression, or a block of code can be repeated a number of times using a loop construct.

Parallel Programs

The target of the Handel-C compiler is low-level hardware. This means that you get massive performance benefits by using parallelism. It is essential for writing efficient programs to instruct the compiler to build hardware to execute statements in parallel.

Handel-C parallelism is true parallelism, not the time-sliced parallelism familiar from general-purpose computers. When instructed to execute two instructions in parallel, those two instructions will be executed at exactly the same instant in time by two separate pieces of hardware.

When a parallel block is encountered, execution flow splits at the start of the parallel block and each branch of the block executes simultaneously. Execution flow then re-joins at the end of the block when all branches have completed. Any branches that complete early are forced to wait for the slowest branch before continuing.

Channel Communication

Channels provide a link between parallel branches. One parallel branch outputs data onto the channel and the other branch reads data from the channel. Channels also provide synchronization so that a transmission can only be completed when both parties are ready for it. If the transmitter is not ready for the communication then the receiver must wait for it to become ready and vice versa.

Scope and variable sharing

The scope of declarations is based around code blocks. A code block is denoted with {...} .This means that:

- Global variables must be declared outside all code blocks.
- An identifier is in scope within a code block and any sub-blocks of that block.
- The scope of the variables is illustrated below:

```
int w;
void main(void)
{
    int x;
    {
```

```
                int y;
        }
        {
                int z;
        }
}
```

Since parallel constructs are simply code blocks, variables can be in scope in two parallel branches of code. This can lead to resource conflicts if the variable is written to simultaneously by more than one of the branches. Handel-C states that a single variable must not be written to by more than one parallel branch but may be read from by several parallel branches.

If you wish to write to the same variable from several processes, the correct way to do so is by using channels that are read from in a single process. This process can use a "prialt" statement to select which channel is ready to be read from first, and that channel is the only one that will be allowed to write to the variable.

```
while(1)
prialt
{
case chan1 ? y:
break;
case chan2 ? y:
break
case chan3 ? y:
break;
}
```

In this case, three separate processes can attempt to change the value of y by sending data down the channels, chan1, chan2 and chan3. y will be changed by whichever process sends the data first.

Handel-C uses much of the syntax of conventional C with the addition of inherent parallelism. You can write sequential programs in Handel-C, but to gain maximum benefit in performance from the target hardware you must use its parallel constructs. These may be new to some users. If you are familiar with conventional C you will recognize nearly all the other features.

7. Experimental Results

Evolving Neural Network Modules on a PC and on the Celoxica Board

The aim of the experiment described in this section is to evolve a "GenNet", i.e. a Genetically Programmed Neural Net that outputs a desired signal. The experiment was conducted both on a PC and on the Celoxica board. Based on the two evolution times of the GenNet using the two methods, a comparison was made to see how great a speed-up of the evolution can be achieved by using the Celoxica Board compared to using a software based evolution on the PC. The GenNet evolution program that was

2.6. MOSES in Action

Let's go through all of the steps that are needed to apply MOSES to a small problem, the artificial ant on the Santa Fe trail [10], and describe the search process. The artificial ant domain is a two-dimensional grid landscape where each cell may or may not contain a piece of food. The artificial ant has a location (a cell) and orientation (facing up, down, left, or right), and navigates the landscape via a primitive sensor, which detects whether or not there is food in the cell that the ant is facing, and primitive actuators *move* (take a single step forward), *right* (rotate 90 degrees clockwise), and *left* (rotate 90 degrees counter-clockwise). The Santa Fe trail problem is a particular 32 x 32 toroidal grid with food scattered on it (Figure 6), and a fitness function counting the number of unique pieces of food the ant eats (by entering the cell containing the food) within 600 steps (movement and 90 degree rotations are considered single steps).

Programs are composed of the primitive actions taking no arguments, a conditional (*if-food-ahead*),[9] which takes two arguments and evaluates one or the other based on whether or not there is food ahead, and *progn*, which takes a variable number of arguments and sequentially evaluates all of them from left to right. To score a program, it is evaluated continuously until 600 time steps have passed, or all of the food is eaten (whichever comes first). Thus for example, the program *if-food-ahead(m, r)* moves forward as long as there is food ahead of it, at which point it rotates clockwise until food is again spotted. It's can successfully navigate the first two turns of the Santa Fe trail, but cannot cross "gaps" in the trail, giving it a final score of 11.

The first step in applying MOSES is to decide what our reduction rules should look like. This program space has several clear sources of redundancy leading to over-representation that we can eliminate, leading to the following reduction rules:

1. Any sequence of rotations may be reduced to either a left rotation, a right rotation, or a reversal, for example:

   progn(left, left, left)
   reduces to
   right

2. Any *if-food-ahead* statement which is the child of an *if-food-ahead* statement may be eliminated, as one of its branches is clearly irrelevant, for example:

   if-food-ahead(m, if-food-ahead(l, r))
   reduces to
   if-food-ahead(m, r)

3. Any *progn* statement which is the child of a *progn* statement may be eliminated and replaced by its children, for example:

   progn(progn(left, move), move)
   reduces to
   progn(left, move, move)

The representation language for the ant problem is simple enough that these are the only three rules needed – in principle there could be many more. The first rule may be seen as a consequence of general domain-knowledge pertaining to rotation. The second

[9] This formulation is equivalent to using a general three-argument *if-then-else* statement with a predicate as the first argument, as there is only a single predicate (*food-ahead*) for the ant problem.

run on the PC was written in the “C” language, whereas the program run on the Celoxica board was written using the “Handel-C” language.

Handel-C is a C like language that can program the hardware, whereas C language is used to program the PC. The gate-level netlist output of Celoxica’s “DK” (Designer Kit) software contains basic gates: OR, XOR, NOT and AND. The FPGA board is made of LookUp Tables (LUT, with 4 inputs and 1 output), flip-flops (FF) and other components. The output of the DK, which is the bit file, is downloaded to the board to configure its components. Technology mapping is the process of packing gates into these components. This is very different from traditional programming of a PC. A program in a PC is stored in memory. At each clock tick the CPU will fetch an instruction code and execute it, then fetch another code, etc. The program on the board is just a set of LUTs, so the execution of the program is at electronic speeds.

There are a few problems when coding the GenNet using Handel-C. Doing real number calculations is not as easy as in C. There is no variable type such as float, double etc in Handel-C. The fitness calculation, decoding of the chromosome into weights and chromosome fitness comparison all need real number calculators. Handel-C does provide a floating-point library and a fixed-point library. Float-point calculations will generate LUTs of great depth, which means longer propagation times, and slower running times. Fixed-point library calculations are preferred here because each real number variable will be represented as a sequence of bits of specified length. The first part of the bits will be interpreted as integers, and the second part will be interpreted as decimals. There are signed and unsigned fixed-point variables, while the signed fixed-point uses 2’s complement representation. Attention must be paid too when using fixed-point library calculations. Using too many bits in each variable will increase the number of LUTs, while too few bits in each variable will decrease the precision of the calculation.

Handel-C has many features that are suitable for evolutionary computation. Firstly, the variable in Handel-C can be of any length. Handel-C also provides bit-wise operations. Thus using Handel-C to code chromosomes introduces great flexibility. Secondly, the “par” block provides parallelism, which reduces the running time of decoding each chromosome and the fitness calculation. Thirdly, the program is a set of LUTs on the board, so the running time is at electronic speeds. Fourthly, pipeline techniques can be applied to increase the throughput of some blocks of code.

The Handel-C program is tested and simulated in the PC. It is then placed and routed using Xilinx’s ISE 7.1. Our Celoxica board is a Xilinx Virtex RC203. The GenNet program in C was run on a PC with a 1.69 GHZ Centrino CPU, and 1 GB of memory.

The program was run 10 times each on the board and the PC. Evolution time of each test was recorded. The average evolution time on the PC is 57.4 seconds, while the average evolution time on Celoxica board is 7.1 seconds. ***The speedup of the evolution time on the FPGA board is 8.1 times faster than on the PC.***

This is a great improvement compared with the evolution time on a PC. Although a great speedup is achieved, there are many possible improvements to the current method that can further increase the evolution speed on the board. The experiment on the current board has some limitations. If a larger and newer version of the board (e.g. the RC300) becomes available, results will be much better (i.e. a greater speedup becomes possible).

1. The current GA code’s level of parallelism is not sufficient. Because the number of LUTs (the number of available gates) and Configurable Logic Blocks (CLBs) on the

board is a limited resource (28,672 LUTs for Virtex RC203, 4 slices), the current GA code does not have enough parallel code blocks (e.g. using the "par" feature). Handel-C provides "par" blocks that can be used to express both coarse and fine-grained parallelism. (For example, see the Handel-C tutorial on the Celoxica website [4]). Individual statements and functions can be run in parallel. The more "par" blocks used, the more parallelism is achieved. However, when "par" blocks are mapped into hardware, their parallel execution requires more circuitry, which means more resources (i.e. logic gates, etc) will be consumed. The Celoxica board has a limited number of LUTs, so that not too many "par" blocks can be used. If too many parallel structures are used in the Handel-C code, the final number of LUTs will be too large to fit into the board (RC203) used in the experiment.

2. Decoding the chromosome into the weights, the fitness calculation and mutation can all be done in parallel. This requires many arrays to store temporary variables. Arrays in the Handel-C provide parallel access to all elements, which is suitable for "par" blocks. But too many indexed or two-dimensional arrays used will introduce the same problem, i.e. too many resources will be eaten up because of the multiplexing. Instead, we used block storage of RAM a lot in the code to avoid the problem of insufficient resources on the board. With RAM, at each clock tick, only one element of data can be accessed, which means the code related to the RAM must be sequential. However this decreases the level of parallelism of the GA code.

Our original Handel-C code had many parallel blocks, to such an extent, that it could not fit into the current board available for our experiment. It generated 37% more LUTs than the maximum capacity of the current board. In order to reduce the number of LUTs needed, many pieces of parallel code had to be rewritten. If most of the code in the GA can be parallel, then definitely much faster speedups will be achieved.

3. Pipeline techniques can be applied to the GA code to increase the throughput. Some complex expressions or calculations in the current code can be broken into smaller pieces and run simultaneously. Each stage in the pipeline reads the results of the previous stage while the new value is being written. This will result in increased data latency. But the downside of this approach is it results in increased flip-flop usage. Thus, all the above improvements are only possible when tested on a newer version or bigger board.

Martin performed an empirical study [5] with Genetic Programming using Handel-C and an FPGA board and reported a speed-up factor of over 400 times when compared with a software implementation of the same algorithm, but the PC used was an AMD 200 MHZ CPU and the problem had no real-number calculations. The problem and experimental setup was quite different from our evolution of GenNets because all our GenNet calculations had to been done in fixed-point, which consumed a lot of resources, thus preventing the program from being parallelized and pipelined easily.

8. More Recent Work

In an attempt to increase the 8-fold speed-up factor mentioned above, using the Celoxica board, relative to performing the same task on a PC, we undertook some recent experiments in which we were less constrained to make compromises (e.g. by having to use less PAR statements) than we were when undertaking the experiments discussed in earlier sections. We were less constrained as a consequence of performing simpler experiments that required less Handel-C code, and less logic gates on the Xilinx chip on the Celoxica board. We evolved neural nets that were smaller, having fewer neurons, and connections, and whose evolved tasks were simpler. As a result, we were able to take greater advantage of the intrinsically faster evolution speeds of the Celoxica board, and confirmed our suspicion that approximately a 50-fold speed up would be possible, and indeed it was.

The first of these experiments (of 2 so far) was to evolve a fully connected neural network of only 3 neurons, hence 9 connections, whose binary fraction weights had only 4 bits to represent them (plus one bit for the sign +/- of the weight). The task was to output a constant "target" signal (of value corresponding to the binary fraction .1001, whose decimal value = 0.5625). The fitness was defined to be the sum of the squares of the differences between the actual output values and the target value, over 15 clock cycles. The population size was 20, and the number of generations was 16000. The evolution took 47 minutes, 36 seconds on the PC, and 51 seconds on the Celoxica board, ***a speed-up factor of about 56***. Only about 7% of the logic gates on the Xilinx chip of the Celoxica board were used. The Genetic Algorithm took up most of this 7%.

In a similar experiment, we then tried to evolve 4-neuron modules. Most of the parameter values of this second experiment were the same as before. This time the PC evolution time was 67 minutes 20 seconds, and the Celoxica board evolution time was 81 seconds, i.e. ***a speed-up factor of 50***. The percentage of the Xilinx chip occupied by the Handel-C program was approximately 223,000/3,000,000 approx 7.4%.

These speed-up factors were encouraging. However, we noticed something that dampened our enthusiasm somewhat, and that was the time needed to perform the ***routing*** of the electronic connections in the Xilinx chip of the Celoxica board. Xilinx provides special software to perform the routing of the electronic wiring that connects the logic gates on their programmable chip. We found that this routing time for our second experiment was about ***40 minutes.***

However, the whole point of using the Celoxica board is to accelerate the evolution of individual neural net modules, so that it becomes practical to evolve 10,000s of them to build artificial brains. If it takes about 1 hour (let's say) to evolve a neural net module on a PC, then what is the great advantage of using a Celoxica board to do the same thing, if the routing time is almost as long as the PC evolution time, even if the actual execution time of the routed circuit performs 50 times faster?!

This seems like a crushing objection to our overall approach, but there is a loophole fortunately, which is that one evolves a "generic" neural net that is then fed data into the circuit already routed on the Xilinx chip. In other words, the routing of the circuitry on the programmable chip occurs once, and is slow, but once the routing is done, the resulting circuit can be used many times over, in the sense that different neural net modules can be evolved by sending in different data to the same circuit. Thus one does not have to route each neural net module that is evolved in the chip. Sending in the data to the chip takes only seconds, compared to about 40 minutes to route the chip. Hence, we can take advantage of the much greater electronic speed of the Xilinx chip on the Celoxica board. Future work along these lines is currently being thought about and planned. See the next section on future ideas for further details.

9. Future Ideas

The immediate future work to be undertaken comprises two tasks. The FPGA on our Celoxica board has only 3 mega-gates. The Celoxica company field engineers, have very recently offered us to run our Handel-C code on one of their boards with a bigger chip (i.e. one with 6 mega-gates), so that our Handel-C code in our first experiment that originally had too many "par" statements can fit better into the larger chip, and hence run faster. This experiment is something we need to do.

Another immediate task is to translate the Handel-C code that we have already written into a more pipelined approach. Pipelining, a well-known technique in computer science, requires extra silicon (extra logic gates) but is faster to operate, due to its greater parallelism. We can send off this pipelined code to Celoxica also, or perhaps buy our own bigger FPGA board. This type of reasoning will remain valid as long as Moore's Law remains valid. We can expect every year or so, to be able to evolve neural net modules faster, due to larger FPGAs.

Greater speedups will allow neural net modules to be evolved considerably faster compared to using software techniques on an ordinary PC, so that a rather complex module containing multiple tests for its evolution (e.g. a module that responds with a strong signal if it sees any one of the 5 letters a, b, c, d, e, but a weak signal if it sees any of the other 21 letters, will require 26 "tests") that would take many hours on a PC, could be evolved in minutes using the Celoxica board approach.

Such speedups will revolutionize brain building. It would then become practical to evolve 10,000s of neural net modules in a time that would be practical within human patience levels. A relatively small brain building team (say of 10 people) could then build an artificial brain (of 10,000 modules) in 5 months, at a rate of 10 evolved modules per working day per person.

If the total speedup can be made closer to 1000 rather than 100, then it is conceivable that new multi-module evolutionary algorithms could be created, that would make use of this speedup to evolve not only the intra-module weights, but the inter-module connections as well. One of the weaknesses with the authors' current brain building approach is that the modules are evolved individually, irrespective of how they will interact with other modules in a network of modules. Very rapid evolution of individual modules would allow new approaches to multi-module evolution.

10. Summary

This chapter has presented results showing that a Celoxica FPGA board is capable of speeding up the evolution of neural network modules by a factor of about 10 to 50 times, depending on the size of the neural net being evolved. In the case of the first experiment with a large neural net (e.g. 20 neurons), which achieved only an 8-fold speedup, the possibility still exists of producing a speedup of perhaps up to 50 times using larger chips and pipelining techniques - an essential step if artificial brains, comprised of 10,000s of such modules are to be evolved in a reasonable time, and then run in real time in interconnected form in an ordinary PC. At the present time, the evolution of a single neural network can take many hours, a fact that makes brain building according to our PC-based approach quite impractical.

Smaller Gas

Most of the gates (flip flops) on the Xilinx chip on the Celoxica board, were taken up by the Genetic Algorithm. With the 3 and 4 neuron network experiments we tried, only about 7% of the gates were used. This is encouraging. We will try to evolve larger modules, i.e. with a larger number of neurons, and hence connections. The number of connections grows as the square of the number of neurons. There are also smaller GAs in the literature, usually called "Compact Genetic Algorithms" (CGAs) that by definition, are very simple and hence need fewer logic gates for their electronic implementation. We may be able to evolve electronically, larger modules with small GAs, and hence really push up the size of the modules we can evolve (with a 50-fold speedup).

Another factor assisting the growth in the size of modules is of course Moore's Law. For example, the next generation Celoxica board, beyond the RC203 that we are currently using, has a Xilinx FPGA chip that contains 6 million logic gates, i.e. a doubling compared to our RC203 Celoxica board. Celoxica does indeed have a new board based on the new Xilinx chip"Virtex 4". So, our next research project will be to see how large our neural modules can become, i.e. just how many fully connected neurons can be evolved electronically on the Celoxica board? More interestingly is the question, "How many more years of Moore's Law will be needed before it will be possible to evolve electronically, neural net modules of reasonable size (i.e. having about 12-20 neurons)? It looks as though the answer is only a few years away (or none at all?). Once such powerful modules are readily evolvable, then the real work of building artificial brains can begin. The underlying technology, i.e. the electronic evolution of large numbers of neural net modules, will make the production of 10,000s of evolved modules needed to build an artificial brain, practical. The real challenge then of designing an artificial brain can then begin, and result hopefully in the creation a new research field, namely "Brain Building" or "Artificial Brains".

As mentioned at the end of section 9, one other research challenge remaining is how to design a "generically evolvable" neural net to overcome the slow routing problem. For example, one evolves the generic circuit *once*, and then sends in different fitness definitions as external data from the PC to the circuit. To change the fitness definition, simply means changing the data that is input to the "changeless" generic circuit. Fleshing out the details of these initial ideas remains a future research topic.

Postscript : Answering a Reviewer

This postscript contains questions from a reviewer to an earlier version of this chapter. We thought some of the questions were interesting, so we include them here, with our answers. The questions are in italic, beginning with a Q, and our answers begin with the letter A.

Q : This is a very interesting chapter that presents results that seem to indicate that computational power is not going to be a major obstacle to creating generally intelligent computers in the mid-term.

A: We think computational power is a necessary condition at least. Modern electronics is already capable of creating artificial brains with tens of thousands of evolved neural net modules. Now that it is possible, the authors would like to actually do it.

Q: Some questions about comparisons with more conventional hardware. A quad-core processor on a four-processor machine could in theory give a 16x speedup over a single processor. If that turns out to be approximately feasible, what will the cost-benefit tradeoff be between this approach and the approach of using conventional hardware? Will the underlying FPGA speed grow as fast as processor speed? If not, then Moore's law could potentially obviate this approach.

A: Special hardware, such as a quad core processor and the like, are definitely viable alternatives to what we propose in this chapter, but are more expensive. Our Celoxica board costs only about $1000, so seems a cheap way to get a 50-fold speedup over an ordinary PC. If the speeds of future FPGA boards do not grow faster than the speeds of PCs, then that would be something we would welcome. The FPGA board is only a tool for us to accelerate the evolution speed of the neural net modules. If a faster cheaper way can be found, that's fine by us.

Q: The authors seem to suggest that only real barrier to achieving general intelligence is affordable hardware speed. Do they really believe that all the intellectual problems have been solved?

A: If we appear to be suggesting that, then that is not our intention. Of course, affordable hardware speed is only the initial and necessary condition to achieving general intelligence. Once the hardware speed is achieved, then our real research challenge of designing interesting capable artificial brains can begin, with the emphasis on the word *begin*.

Q: There are many other approaches to hardware acceleration for neural systems. Could the authors say a bit about how their approach compares?

A: There are approaches such as ASIC chips (i.e. custom designed chips for specific tasks). Of course the great advantage of the FPGA approach is it flexibility, its reprogrammability. If we want to evolve a different neural net model in the FPGA, we can do that easily. With a non reprogrammable approach we couldn't. ASICS may be faster than FPGAs, but their flexibility far out ways their relatively slower speed relative to ASICS, we feel.

Postscript : Last Minute Results

As proof that the work of this chapter is ongoing, we report here some very recent results concerning two experiments. The first was to see how many neurons N we could fit into the FPGA chip, using a more compact GA (i.e. the CGA, "Compact GA" mentioned in section 10 (Summary). By using fewer lines of code to specify the GA, as is the case in a CGA, there was a lot more room on the FPGA for more neurons N in the neural network. For the same experiment mentioned in section 9, using the CGA, we were able to place N = 28 neurons on the chip. This is far more than we need, so we undertook a more demanding experiment. The aim this time was to evolve (using the CGA mentioned above) a sine curve output for half a wavelength. The number of ticks of the clock used for the curve (i.e. one tick is one cycle of calculating the output signal for each neuron in the network) was 45. The number of bits in the weights was increased to 8 (i.e. 1 for the sign, and 7 for the weight value). The fitness definition was the sum of the squares of the errors between the target sine half curve (i.e. y(t) = sin(pi*t/45)), and the actual output signals over the 45 ticks t. The number of neurons as 12, population size was 256, bit string chromosome length was 12*12*8 = 1152 bits. The number of generations used was 128,000. This half sine curve evolved well, showing that a non trivial neural net could be evolved in the 3M gates available in the

FPGA of the Celoxica board. The speedup factor was about 57 times compared to evolving the same task on the same PC used to control the Celoxica board.

In the near future, we will evolve many other single neural network modules using our Celoxica board, to show that they can be evolved. The next immediate task is to create an approach we call "Generic Evolution". By this we mean creating a generic evolvable model, that is routed once (or only a few times) in the FPGA and then used multiple times, by sending in data signals to the FPGA. Each data set that goes into the chip is used to evolve a neural net module. This data contains such things as the number of positive and negative examples of training vectors for the neural net evolution, and the training vectors themselves. The model expects data input to have a given dimensionality, e.g. 1 dimensional bit strings, or 2D pixel grid input from a digital camera, etc. We are currently working on this, to overcome the problem of having to route the FPGA for each module evolution. Since the routing takes about 40 minutes for a full chip, this is too slow. So by having a generic model routed once only in the chip, we can send it different data for different neural net module evolutions. Sending in data to the already routed chip takes only a few seconds.

As a more concrete illustration of this "generic evolution" idea, we provide the following example. Assume we want to evolve several hundred 2D pattern recognition neural net modules. The pattern to be detected is "shone" onto an 8 by 8 pixel grid of photo-cell detectors, each of whose signal strength outputs is strong if strong light falls on the photo-cell, and is weak if the light falling on it is weak. These 64 light intensity output signals are fed into a fully connected 16 neuron neural network. Hence each neuron receives 4 external signals from the pixel grid. If the pattern on the grid is of the type that the neural net module has been evolved to detect, then the module will output a strong signal, otherwise a weak signal. To evolve such a detector module, we create a set of positive examples of the pattern, e.g. a "vowel" detector (i.e. the letters A,E,I,O,U). Similarly, we create a set of negative examples (e.g. the 21 consonants). We then shine the 5 vowels onto the grid, say for 100 ticks each. (One tick is defined to be the time needed for all neurons in the module to calculate their output signals.) We then shine the 21 consonants onto the grid for 100 ticks each.

The 64 (8 by 8) pixel values from the photocell grid are converted into a 1 dimensional (64 element) vector, by concatenating the 8 rows of the grid. There will be 26 (5 + 21) such vectors. Thus the data to be sent to the generic circuit will take the following form :- (N1, i.e. the number of positive examples, N2, i.e. the number of negative examples, C, i.e. the number of clock ticks per example, and then the (N1 + N2) 64-element input vectors.

The Handel-C code is written such that it expects the above parameters to be sent to it once the routing of the code has been completed. One can evolve hundreds of different 2D pattern detection neural net modules, in this way, without having to reroute the chip for each module evolution. The various modules have generic features, e.g. they all use a 16 neuron neural network, with a 64 input signal. The N1 positive examples are input first, then the N2 negative examples. The fitness definition of the neural net module is also generic. The target (i.e. desired) output signal of the evolving neural net module is high if a positive example is input, and low if a negative example is input. Hence the fitness definition can be generalized, e.g. to be the inverse of the sum of the squares of the differences between the target signal values and the actual signal values for C*(N1+N2) ticks. This fitness definition will be a function of the various parameters (N1, N2, C, etc) that are sent in as data for each neural net module evolution. Hence this generic fitness definition need only be coded and compiled and routed once for the evolution of hundreds of 2D pattern recognition neural net modules.

If the generic fitness definition changes (e.g. by having a new neural net model, or different input vector formats) then another routing can be performed, costing 40 minutes. But in practice, much time is saved by using this “generic evolution” approach that is new, and an invention of our research team. It is becoming an important theme in our use of the Celoxica board to accelerate the evolution of tens of thousands of neural net modules to build artificial brains.

References

[1] Hugo de Garis, Michael Korkin, THE CAM-BRAIN MACHINE (CBM) An FPGA Based Hardware Tool which Evolves a 1000 Neuron Net Circuit Module in Seconds and Updates a 75 Million Neuron Artificial Brain for Real Time Robot Control, Neurocomputing journal, Elsevier, Vol. 42, Issue 1-4, February, 2002. Special issue on Evolutionary Neural Systems, guest editor: Prof. Hugo de Garis. Downloadable at http://www.iss.whu.edu.cn/degaris/papers
[2] www.celoxica.com
[3] See http://www.iss.whu.edu.cn/degaris/coursestaught.htm Click on the CS7910 course.
[4] www.celoxica.com
[5] Peter N. Martin, Genetic Programming in Hardware, PhD thesis, University of Essex, 2003, http://homepage.ntlworld.com/petemartin/HardwareGeneticProgramming.pdf

Advances in Artificial General Intelligence: Concepts, Architectures and Algorithms
B. Goertzel and P. Wang (Eds.)
IOS Press, 2007
© 2007 The authors and IOS Press. All rights reserved.

Complex Systems, Artificial Intelligence and Theoretical Psychology

Richard LOOSEMORE
Surfing Samurai Robots, Genoa NY, USA
rloosemore@surfingsamurairobots.com

Abstract. The main finding of complex systems research is that there can be a disconnect between the local behavior of the interacting elements of a complex system and regularities that are observed in the global behavior of that system, making it virtually impossible to derive the global behavior from the local rules. It is arguable that intelligent systems must involve some amount of complexity, and so the global behavior of AI systems would therefore not be expected to have an analytic relation to their constituent mechanisms. This has serious implications for the methodology of AI. This paper suggests that AI researchers move toward a more empirical research paradigm, referred to as "theoretical psychology," in which systematic experimentation is used to discover how the putative local mechanisms of intelligence relate to their global performance. There are reasons to expect that this new approach may allow AI to escape from a trap that has dogged it for much of its history: on the few previous occasions that something similar has been tried, the results were both impressive and quick to arrive.

Keywords. Complex Systems, Artificial Intelligence, Adaptation, Self-Organizing Systems, Cognition, Methodology, Theoretical Psychology, Cognitive Science.

Introduction

One the most basic assumptions made by Artificial Intelligence researchers is that the overall behavior of an AI system is related in a lawful, comprehensible way to the low-level mechanisms that drive the system.

I am going to argue that this apparently innocent assumption is broken, because all intelligent systems, regardless of how they are designed, must be complex systems[1], and the definition of a complex system is that the overall behavior of the system is not always related in a comprehensible way to the low level mechanisms that cause the behavior. I will further argue that the consequences of this assumption being broken are so serious that the current approaches to AI will never lead to a full, human-level artificial intelligence. The good news is that there is a way to solve the problem, and this solution could unlock the floodgates of progress in AI. The bad news is that many AI researchers are so uncomfortable with this solution that they are determined to resist it at all costs. My goal in this paper is to discuss the problem, outline the solution, and try to raise the awkward issues surrounding the negative reaction that the solution tends to provoke.

1 Unless otherwise noted, the terms "complex" and "complexity" will always be used here to refer to aspects of complex systems, not to the general sense that means "complicated," nor to the mathematical analysis of algorithmic complexity.

This is the "Complex Systems Problem," and in the second part of the paper I propose a new methodology designed to overcome it. The new methodology involves a combination of three factors: (i) a grassroots integration of AI and the field of cognitive psychology, (ii) a "complete framework first" approach to AI system design, and (iii) the use of programs of systematic experimentation, in which large numbers of systems are built in order to discover (rather than assume) the relationships between global behavior and local mechanisms.

It is important to emphasize that the proposed solution is rather more than just a new approach to AI. It is based on a body of knowledge (about human cognition) and a philosophical attitude (about the unavoidable need for empirical science) that is categorically rejected by many of the researchers who now dominate the AI field. The awkward truth is that the proposed new methodology has little use for most of the work that has been done in the AI field in the last two decades, and even less use for the devotion to formal systems that seems indispensible to many in the AI community.

These considerations about the social psychology of research in AI are important because they underline the extraordinary seriousness of the Complex Systems Problem. If the only way to solve the problem is to declare the personal philosophy and expertise of many AI researchers to be irrelevant at best, and obstructive at worst, the problem is unlikely to even be acknowledged by the community, let alone addressed.

The ultimate conclusion of this paper is that the Complex Systems Problem is the single biggest reason why a human-level artificial intelligence has not been built in the last fifty years, but that if someone has the vision and resolve to fly in the face of orthodoxy and implement the solution proposed here, the results could be dramatic. Quite apart from the fact that this approach has never been tried before, there are positive reasons to believe that it would succeed where conventional AI has not.

1. Complex Systems

This section and the next examine the nature of complex systems and their impact on AI research.

A complex system is one in which the local interactions between the components of the system lead to regularities in the overall, global behavior of the system that appear to be impossible to derive in a rigorous, analytic way from knowledge of the local interactions.

This definition is part empirical fact, part mathematical intuition. As empirical fact, it is a summary of observations made independently by a number of researchers who tried to understand the behavior of systems in a wide variety of fields [1].

The mathematical intuition aspect is perhaps more powerful, because it derives from something that has been said by nonlinear mathematicians for a long time: taken as a whole, the space of all possible mathematical systems contains a vast, unexplored region in which the systems are easy enough to define, but seem to be beyond the reach of analytic solution. It would probably be no exaggeration to say that the part of this space that contains the sum total of all systems that have so far been rigorously analyzed is considered by many to be an unthinkably small corner of the whole space. The crucial part of the intuition concerns how far the tractable corner of this space might expand in the future: if human mathematicians could work for an arbitrary length of time, would they be able to find analytic solutions for most of the systems in the unexplored region? There is no particular reason to assume that we could do this, and it would not be surprising if there exist no analytic solutions whatsoever that describe the behavior of almost all of the systems in that space.

The implication of this intuition is that if a system has components that interact in such a nonlinear, tangled way that we have doubts about whether any analytic understanding of its behavior will ever be possible, we should at least adopt the precautionary principle that there might not be any analytic explanation of the system. We should be especially careful when we observe regularities in the global behavior of such systems: those regularities should not be taken as a clue that a formal, analytic explanation is lurking beneath the surface. The term "complex system" is used to describe precisely those cases where the global behavior of the system shows interesting regularities, and is not completely random, but where the nature of the interactions between the components is such that we would normally expect the consequences of those interactions to be beyond the reach of analytic solutions.

This defining feature of complexity is sometimes referred to as "emergence" [2], but since that term has wider significance (and is even taken by some to have vitalist connotations that border on the metaphysical), the term "Global-Local Disconnect" (or GLD) will be used here. The GLD merely signifies that it might be difficult or impossible to derive analytic explanations of global regularities that we observe in the system, given only a knowledge of the local rules that drive the system.

1.1. The Recipe for Complexity

Speaking in purely empirical terms, it is possible to give a list of design ingredients that tend to make a system complex:

- The system contains large numbers of similar computational elements.
- Simple rules govern the interactions between elements.
- There is a significant degree of nonlinearity in the element interactions.
- There is adaptation (sensitivity to history) on the part of the elements.
- There is sensitivity to an external environment.

When the above features are present and the system parameters are chosen so that system activity does not go into a locked-up state where nothing changes, and the activity does not go into an infinite loop, then there is a high probability that the system will show signs of complexity.

It should be understood, however, that this recipe only gives a flavor of what it takes to be complex: it is possible to find systems that lack some of these features but which would still be considered complex.

1.2. The Absence of a Clear Diagnostic

One fact about complex systems is especially important in the context of the present paper, where a great deal hinges on whether or not AI systems can be proven to have a significant amount of complexity in them:

- It is (virtually) impossible to find a compact diagnostic test that can be used to separate complex from non-complex systems, because the property of "being a complex system" is itself one of those global regularities that, if the system is complex, cannot be derived analytically from the local features of the system.

What this means is that any debate about whether or not intelligent systems are complex must not involve a demand for a definitive proof or test of complexity, because there is no such thing.

The "virtually" qualifier, above, refers to the fact that complex systems are not excluded from having global properties that are derivable from local features or mechanisms: it is just that such exceptions to the GLD are unusual, especially where, as in this case, the property is supposed to hold across a large, nonhomogeneous class of systems.

1.3. Controversy

The scientific status of complex systems research has been disputed. Skeptics like John Horgan [3] have argued that the term "complex system" means whatever people want it to mean, and that in spite of all the effort put into the field there is little in the way of a theory that might unify the disparate phenomena under study.

This criticism may have some merit, but it is important to understand that Horgan's points have no relevance to the arguments presented here. It is not important that there be a general theory of complexity, because all that matters here is one rather simple observation about the behavior of complex systems: the global-local disconnect. The GLD is a deceptively innocent idea—it seems to lack the kind of weight needed to disrupt an entire research program—but if it is real, and if complexity plays a significant role in the behavior of intelligent systems, then much of mainstream AI research may nevertheless have been severely compromised by it.

1.4. Computational Irreducibility

Could it be that complex systems only appear to have a global-local disconnect because current mathematical tools are not up to the task of analyzing them? Perhaps in the fullness of time these systems will be understood in such a way that the global regularities are derivable from the local mechanisms? In that case, the GLD might be nothing more than temporary pessimism.

This flies in the face of the intent of the complex systems idea, which is that something more than mere pessimism is involved; there really is something new and fundamentally different about many of these systems, and there will not be any future mathematical formalism that eliminates the GLD.

Stephen Wolfram has used the term "computational irreducibility" [4] to capture one version of this idea. Physical systems obey laws that govern their behavior, and Wolfram's suggestion is that those laws can be viewed as if they were computations being carried out by a physical system in order to calculate the next set of values for its physical state. What we have known ever since the time of Isaac Newton is that we can reduce these laws to equations that allow us to short-circuit the physical computations and predict what a physical system will do before it does it. If we want to know where a planet will be next week, we can use equations that do not take a whole week to complete, whereas the computations that the planet is effectively executing when it follows its local laws of nature do take a week.

What Wolfram points out is that although we have had great success in finding mathematical systems that allow us to reduce the behavior of physical systems in this way, there is no guarantee that all physical systems should be amenable to this trick, and that, on the contrary, there appears to be a very large class of systems that will never be so reduced. Running a simulation of such a system would then be the only viable way to find out how it behaves.

Can this idea of computational irreducibility be proved? Almost certainly not: at this stage it is an empirical observation about the nature of the world, and there may never be any way to formally prove that a given system is permanently irreducible. It has the status of a meta-law of the universe: a law that says that not all the regularities in the universe can be described with equations that can be solved.

The main implication of computational irreducibility is that our faith in the power of mathematical descriptions of the physical world is misplaced: a system can show regularities in its overall behavior without there being any analytic explanation that connects those regularities to local mechanisms.

2. The Relationship Between Complex Systems and AI

One of the main arguments advanced in this paper is that complexity can be present in AI systems in a subtle way. This is in contrast to the widespread notion that the opposite is true: that those advocating the idea that intelligence involves complexity are trying to assert that intelligent behavior should be a floridly emergent property of systems in which there is no relationship whatsoever between the system components and the overall behavior.

While there may be some who advocate such an extreme-emergence agenda, that is certainly not what is proposed here. It is simply not true, in general, that complexity needs to make itself felt in a dramatic way. Specifically, what is claimed here is that complexity can be quiet and unobtrusive, while at the same time having a significant impact on the overall behavior of an intelligent system.

2.1. Conway's Game of Life

One way to see this quiet-but-significant effect is to look at a concrete example. This section is therefore centered on a set of variations on one of the most elementary of complex systems: John Horton Conway's cellular automaton called "Life" [5].

Life is a cellular automaton based on a 2-dimensional square grid of cells, with changes happening simultaneously to all of the cells at every tick of a global clock. Each cell can be either ON or OFF, and the rules for what the cell should do after the next tick depend only on its own state and the state of its eight nearest neighbors at the current time:

- If the cell is currently ON, stay ON if there are 2 or 3 neighbors ON, else go to the OFF state.
- If the cell is currently OFF, switch to the ON state if 3 neighbors are ON, else stay in the OFF state.

As is now well known, Life displays many patterns of activity that are stable in the sense that there is a period (a number of ticks) after which a pattern repeats exactly. There are also patterns that are not periodic, but which are salient for some reason. An entire zoo of "creatures" has been discovered in this miniature universe, including Gliders that move at a regular speed across the plane, Glider Guns that make Gliders, and Fuses that burn down like real-world fuses. These creatures are clearly an example of observed global regularities in the behavior of the system.

The most interesting question about Life concerns the explanatory relationship between the creatures and the local rules: is it possible to write down a formula that, for example, would predict all of the creatures that exist in Life, given only the definition of the local rules? The GLD says that this is not feasible: there is likely to be no analytic relationship between the local rules and the observed patterns.

2.2. Dynamic Complexity and Static Complexity

Conway's Life can be used to illustrate an important distinction between two different interpretations of how complexity manifests itself: what might be called "dynamic" regularities and "static" regularities. This distinction has to do with the fact that there is more than one level at which global regularities can appear in a complex system. The obvious level is where patterns of cell activation are noticed by us because they are stable. These are dynamic regularities because they occur in the moment-to-moment functioning of the system.

However, a second type of regularity can be seen in the search activity that Conway and his colleagues undertook when they were trying to find a good set of rules to define the game. Conway began with a desired global feature of his target system: broadly speaking, what he wanted was a high value for a parameter that we can refer to as the "generative capacity" (or "G") of the system—he wanted to have comparable numbers of ON cells being created and destroyed. According to one account, he found a set of rules with the desired global property,

> " ... only after the rejection of many patterns, triangular and hexagonal lattices as well as square ones, and of many other laws of birth and death, including the introduction of two and even three sexes. Acres of squared paper were covered, and he and his admiring entourage of graduate students shuffled poker chips, foreign coins, cowrie shells, Go stones or whatever came to hand, until there was a viable balance between life and death." [6]

Why did he not do some mathematical analysis and construct a function to predict the generative capacity from a given system design? Because such a function would have been extremely hard to derive, especially given the wide selection of grid types, gender valencies and rule structures under consideration. Complex systems theorists would now say that such an analytic function is practically impossible.

Although this "generative capacity" parameter is certainly a global regularity, it is a far less obvious type of regularity than the moving patterns of cell activation. To the extent that the generative capacity is not derivable from the local rules and architecture of the system, this is a bona fide example of complexity—but it is both static and more global than the dynamic complexity of the patterns. Static regularities are about characteristics of the system that are to some extent time-independent: whereas a dynamic pattern comes and goes, a parameter like the generative capacity has to be measured over a long period of time, and refers to the system and all of the patterns that occur in it. So there are two senses of "global" at work here: global structures that are larger than single cells, but which can be recognized in a modest amount of time, and global characteristics that encompass the behavior of the system and all of its patterns.

2.3. Searching for Static Regularities

This distinction is important because in some complex systems the dynamic regularities—the equivalent of the creatures in Life—may be the ones that grab all the limelight because of their salience, whereas there may be other, more subtle and almost invisible static regularities that turn out to be the most important features of the entire system.

We can see this point more graphically if we consider what might have happened if Conway had searched for systems characterized by something more ambitious than a high value for the generative capacity. At a very simple level, for example, he could have looked for systems that generate large numbers of interesting creatures, rather than just a good balance between cell creation and destruction. Note that this new concept of "creature density" is a more subjective characteristic, but this subjectivity would not stop us from searching for systems with a high value for the creature density.

Would creature density be classified as a global regularity that is hard to predict from the local rules that govern the system? This is certainly possible, and empirical experience with large numbers of other complex systems indicates that it is likely. Is creature density less valid as a global regularity because it is more subjective? Not at all.

Now imagine a more ambitious goal: a system with a definite boundary, and with input patterns arriving at that boundary from outside, together with the requirement that the system should respond to each external pattern by producing an internal response that is unique in some way. For every pattern appearing at the boundary, one response pattern would have to be generated inside the system.

A further requirement might be that these response patterns be learned over time, and stand in a one-to-one correspondence with the input patterns, so that the internal response could be used to identify a previously seen input pattern.

At an even more ambitious level, consider systems that do all of the above, but which also seem to collect together entire classes of subjectively "similar" boundary patterns, in such a way that all patterns in the class trigger the same internal pattern. (The attentive reader will recognize that this has already been done: unsupervised neural nets can discover limited kinds of pattern in this way [7]).

Finally, imagine a system in which multiple external patterns are allowed to impinge simultaneously on different parts of the boundary, with patterns standing in various "relationships" to one another, and with the requirement that the system produce internal patterns that encode, not just classes of similar patterns, but meta-patterns that are "arrangements" of basic-level patterns (where an arrangement is a group of pattern-instances standing in a certain relationship to one another)—as well as patterns of these higher level patterns, and patterns of those second-level patterns, and so on, without limit.

We have clearly progressed well beyond Conway's modest goal of finding systems that maximize a global parameter called "generative capacity," so maybe it is time to invent a new term to parameterize the degree to which a system exhibits the particular, rather exotic type of global regularity described in the last couple of paragraphs. Call this parameter the "Lintelligence" of the system.

Just as it was for the case of the creature density parameter, we would expect this new parameter to be a global, static regularity, and we would also have no reason to treat it as a less meaningful concept just because it was subjectively defined.

Now imagine that we are looking at an example of a High-Lintelligence cellular automaton. What attracts our attention (perhaps) are the dynamic patterns of activated cells, and if we were interested in complexity we might be impressed by the way that these global structures could not be derived from the local mechanisms. However, if we are interested instead in the overall Lintelligence of the system, the local patterns are a distraction, because the Lintelligence is just as underivable from patterns as the patterns are from the substrate.

The name “Lintelligence” was designed to be suggestive, of course. Could it be that the thing we colloquially call “intelligence” is a global, static regularity like the one just described? Not as simple as Lintelligence, but perhaps an even more elaborate and exotic extension of it. This seems entirely reasonable—and if true, it would mean that our search for systems that exhibit intelligence is a search for systems that (a) have a characteristic that we may never be able to define precisely, and (b) exhibit a global-local-disconnect between intelligence and the local mechanisms that cause it.

2.4. Defining Intelligence

It might be worth noting one implication of this view of what intelligence is. If intelligence is a global, static regularity, we would not expect there to be any compact definition of it. This is consistent with the fact that people find it extremely hard to pin down what intelligence is. Common usage seems to indicate that intelligence is just the cooperative activity of a cluster of mechanisms, that there is no single choice of this set of mechanisms (so there could be many varieties of intelligence), and that there are graded degrees of intelligence, possibly forming a continuum all the way from the simplest thermostat to the human genius level and beyond.

The idea that intelligence could be reduced to a compact, non-circular definition in terms of “agents” and “goals,” or that such a definition could be given a rigorous mathematical formalization, would then amount to nothing more than mathematization for its own sake: a *reductio ad absurdum* of the commonsense definition.

2.5. Multiple Levels

In the sequence of hypothetical systems just considered, there was an assumption that the dynamic patterns would always be obviously complex: that the whole system, from top to bottom, would look like a seething cauldron of emergence, in which nothing could be explained by anything going on at a lower level.

This is not the case: large systems can have structure at many levels of description, and some of those levels can seem more complex than others. In the case of the Life automaton, for example, it is possible to step up from the cell level and note that some of the discovered creatures are actually made out of conjunctions of other, smaller creatures. So there is a sense in which, at the creature level, objects can be used as building blocks for larger objects. What is interesting about this higher level of description is that it is more orderly—more mechanism-like and less complex-like—than the lower level, because the components can be used in a fairly predictable way to assemble larger structures.

In general, a complex system does not have to show its complexity at every level of description. There might be a level at which it gives a fair approximation of being a mechanism-like system. That “fair approximation” label is important, of course: any attempt to pretend that Conway's Life is not complex, and that it could be substituted with a facsimile system in which the lowest level was eliminated, leaving the level 2 objects as the base level, would almost certainly fail. In the case of Life we might try to build a facsimile system in which the basic units were creatures that were explicitly coded to interact in the ways that known Life creatures do, but although this system could very well be rigged so as to work like the original for a while, eventually its behavior would diverge from the real thing.

This question of multiple levels of description is relevant to AI because we try to build intelligent systems using ideas gleaned from insights about our own thought processes. We sometimes talk of the basic units of knowledge—concepts or symbols—as if they have little or no internal structure, and as if they exist at the base level of description of our system. This could be wrong: we could be looking at the equivalent of the second level in the Life automaton, therefore seeing nothing more than an approximation of how the real system works. (This argument is not new, of course: it was one of the main motivations cited by the early connectionists [8]).

2.6. Conclusion: Hidden Complexity

Combining the above notion of global, static regularity with the idea that there can be levels of description that are not obviously complex, it seems entirely plausible that the overall intelligence of an AI system could be a global, static regularity that is elusively dependent on complexity in the system, while at the same time there is a level of the system at which the "symbols" interact in ways that appear more mechanism-like than complex-like. If such were the case, and if we took pains to avoid situations in which the complex nature of the symbols would most likely manifest itself, we might stay convinced for a long time that intelligent systems are not significantly complex.

This matter has serious implications for the methodology of AI. The property of "being intelligent" might turn out to be either a mechanism-like property, or it might be a global, static complexity. If it is mechanism-like (in the way that "being fast" is a mechanism-like property of being a car) then all is well with the standard methodology of AI: we can try to find the low level components that need to be combined to yield intelligence. But if intelligence is a global, static complexity, then we may be in the same position as Conway when he was trying to find the local rules that would generate the global, static complexity that he sought. In the very worst case we might be forced to do exactly what he was obliged to do: large numbers of simulations to discover empirically what components need to be put together to make a system intelligent.

If our insights into the thinking process are showing us components that are not at the base level of our system—if the symbols or concepts that we appear to be manipulating are just an intermediate level of description of the system—then we may think we have identified the laws of thought, and that those laws of thought do not involve a significant amount of complexity, but we may be wrong.

At the very least, there is an important issue to be confronted here.

3. Making a Paradigm Choice

This argument is difficult to defend in detail, not because it is intrinsically weak, but because at the heart of the complexity idea is the fact that if the GLD is real, it will be almost impossible to say for sure whether intelligent systems really do involve significant complexity.

In order to definitively show that intelligence involves a significant amount of complexity, we may have no choice but to build many complete AI systems and gradually amass a database of example systems that all seem to have plausible local mechanisms. With this database in hand, we would then have to stand back and ask how successful these systems are, and whether we are seeing a good relationship between the changes we make to the local mechanisms (in order to fix whatever shortcomings we encounter) and the intelligence of the overall system. If, as part of this process, we make gradual progress and finally reach the goal of a full, general purpose AI system that functions in a completely autonomous way, then the story ends happily.

But the complex systems argument presented here says that this will not happen. When we look at the database of example systems, accumulated over the course of perhaps hundreds of years of work (at the present rate of system-building progress), we may find that, for some inexplicable reason, these systems never actually make it to full, autonomous intelligence. It might take a long time to get a database large enough to merit a statistically significant analysis, but eventually we might reach the empirical conclusion that the overall intelligence of the systems does not bear a good relationship to the quality of the local mechanisms in each case.

If we ignore the Complex Systems Problem, this type of empirical effort may be the only way to come to a firm conclusion on the matter.

As an alternative to such a multi-century empirical program, the best we can do to decide whether complexity is important is to look for evidence that the ingredients of complexity are present. In other words we should stop asking for definitive, analytic proof that complexity is a problem (which the complex systems community claim is an impossible goal, if they are right about what complexity is), and instead look at the other way to define complexity: look at the ingredients and see if they are there.

This is fairly straightforward. All we need to do is observe that a symbol system in which (a) the symbols engage in massive numbers of interactions, with (b) extreme nonlinearity everywhere, and (c) with symbols being allowed to develop over time, with (d) copious interaction with an environment, is a system that possesses all the ingredients of complexity listed earlier. On this count, there are very strong grounds for suspicion.

We could also note a couple of pieces of circumstantial evidence. First, on those past occasions when AI researchers embraced the idea of complexity, as in the case of connectionism, they immediately made striking achievements in system performance: simple algorithms like backpropagation had some astonishing early successes [9][10]. Second, we can observe that the one place complexity would most likely show itself is in situations where powerful learning mechanisms are at work, creating new symbols and modifying old ones on the basis of real world input—and yet this is the one area where conventional AI systems have been most reluctant to tread.

3.1. Paradigm Arguments

The problem with citing suggestive or circumstantial evidence in favor of the idea that complexity is both present and playing a significant role in intelligence, is that this is easily dismissed by those who choose to be skeptical.

For this reason, there is little point making further attempts to argue the case: this is simply a paradigm issue, in Kuhn's classic sense of that term [15]. Deciding whether or not the problem is serious enough to merit a change of methodology is, ultimately, a personal decision that each AI researcher needs to make separately. It would be a waste of time to argue about definitive proof of the role of complexity, because the quicksilver nature of complex systems could always be used to cast doubt on such efforts.

In that spirit, the remainder of this paper moves on from the goal of arguing the case for complexity, and instead tries to sketch the first outline of an alternative approach to studying and building cognitive systems.

4. The Theoretical Psychology Approach

The new methodology that I propose is not about random exploration of different designs for a general, human-level AI, it is about collecting data on the global behavior of large numbers of systems, while at the same time remaining as agnostic as possible about the local mechanisms that might give rise to the global characteristics we desire. Rather than lock our sights on one particular approach—logical inference, bayesian nets, genetic algorithms, neural nets, or some hybrid combination of these name-brand approaches—we should organize our work so that we can look at the behavior of large numbers of different approaches in a structured manner.

The second main ingredient of the approach is the use of what we already know about human cognition. For reasons that will be explained shortly, I believe that it is not possible to ignore human cognition when we do AI. This combination of a close relationship with the data of cognitive science, an empirical attitude to the evaluation of intelligent systems, and the lack of any commitment to directly explain human cognition, is what led to the choice of name for this approach: this is, in effect, a form of "theoretical psychology."

4.1. Generalized Connectionism

The roots of the theoretical psychology approach go back to early connectionism. When connectionist ideas first came to prominence, the core principle was about more than just using neuron-like processing units, it was about exploring the properties of parallel, distributed systems, to find out by empirical experiment what they could do. It was also about the "microstructure" of cognition—the idea that symbols need not be just tokens manipulated by an external processor, but could be processors themselves, or even distributed aspects of clusters of processors. This emphasis on open-minded exploration and the rejection of dogmas about what symbols ought to be like, is closely aligned with the approach described here.

Interestingly, as the connectionist movement matured, it started to restrict itself to the study of networks of neurally inspired units with mathematically tractable properties. This shift in emphasis was probably caused by models such as the Boltzmann machine [11] and backpropagation learning [10], in which the network was designed in such a way that mathematical analysis was capable of describing the global behavior.

But if the Complex Systems Problem is valid, this reliance on mathematical tractability would be a mistake, because it restricts the scope of the field to a very small part of the space of possible systems. There is simply no reason why the systems that show intelligent behavior must necessarily have global behaviors that are mathematically tractable (and therefore computationally reducible).

Rather than confine ourselves to systems that happen to have provable global properties, we should take a broad, empirical look at the properties of large numbers of systems, without regard to their tractability.

4.2. The Role of Intuition

If the only technique available to us were a completely random search of the space of possible cognitive systems, the task would be hopeless. The space to be searched is so large that if we had no other information about the target we might as well just try to randomly evolve an intelligent system, and expect to wait a very long time for results.

It is not the case, however, that we have no information about how an intelligence might function: we have some introspective knowledge of the workings of our own minds.

Introspection is what AI researchers have always used as the original source of their algorithms. Whether committed to human-inspired AI or to the normative, rational approach that eschews the need to build systems in a human-like manner, the ideas that are being formalized and implemented today have their roots in the introspections of past thinkers. Even the concept of logical, rational thought was an idea noticed by the ancient Greek philosophers who looked inside themselves and wondered how it was that their thoughts could lead to valid conclusions about the world.

Introspection has had a bad reputation ever since the behaviorists tried to purge it from the psychological sciences, but in truth the aura of contagion that surrounds it is just an accident of the sociology of science. There is nothing wrong with it, if it is used as a source of inspiration for mechanisms that are then systematically explored and evaluated. For this reason, one of the defining features of the theoretical psychology approach is a search for ways to make the back-and-forth between introspective ideas and experimental tests of those ideas as fluid as possible.

The fact that we are cognitive systems ourselves is a good enough reason to hope that the starting points of our systematic investigations will be better than random.

There is another, less tangible, way that intuition can play a role. If we simulate large numbers of systems that differ by marginal changes in system parameters, we have some hope of observing regularities in the mapping between local mechanisms and global system behavior. Complex systems theory does not say that there will be no relationship whatsoever between the low level and the high level (it does not say that a small change in the low level mechanisms will always lead to an irregular, apparently random change in the high level behavior), it only says that there is no analytical relationship. It may well be that when we start to engage in systematic variations of candidate systems, we discover by inspection and observation that certain changes in low level rules cause lawful changes in the overall system behavior. If this happens, we may be able to converge on a workable design for an intelligent system in a surprisingly short amount of time.

The only way to find this out is to do some real science.

4.3. Neural Inspiration

If a grassroots form of connectionism is the way to go, would it also be a good idea to take the concept of neural inspiration more literally and only study systems that are as close as possible to the ones found in the brain?

It might be a mistake to impose this kind of restriction, because of the access to introspection mentioned above. At the conceptual level (as opposed to the neural level) we have some direct, introspective information about what is going on and why, but we have no such internal access to the workings of the mind at the individual neuron level. We simply cannot introspect any aspects of individual neurons, synaptic vesicles or dendrites, not least because we have little ability to report subjective events at the millisecond timescale.

And although we do have some external access to the functioning of the brain, through brain mapping techniques, signal interventions and post-mortem examination, our ability to get fine detail, and to link that detail to the cognitive level, is a subject of fierce debate within the cognitive science community [12]; [13].

4.4. Frameworks and Quasi-Complete Systems

What does it mean to engage in a program of "systematic exploration" of the space of cognitive systems? To be systematic, such a program needs to be unified by a common conceptual framework that is explicit enough that it allows the relationships between systems to be clearly seen.

The idea of a "framework" is that it defines a set of choices for the broad architectural features of the class of cognitive systems that it expresses. For every one of the various mechanisms that we might anticipate being involved in a complete cognitive system, the framework should have something to say about how that mechanism is instantiated, and how it relates to the rest of the system. These specifications should be explicit enough that a complete system could be constructed in accordance with them.

It would also be vital to keep humans out of the loop as much as possible: systems need to be autonomous. If autonomy is not possible, the human involvement should be made explicit (and credibility discounted accordingly).

Any given research project within this framework would involve a particular instantiation of a system consistent with the framework, together with a particular focus on one or more components of that system. The idea would be to make systematic variations of some aspect of the design and look at the effect those changes had on the global system behavior.

Thus, someone determined to show that (for example) logical reasoning was a valid way to build intelligent systems would instantiate a set of traditional-looking symbols, inference mechanisms and all of the symbol-learning mechanisms and sensorimotor apparatus needed to connect the system with its environment. The mechanisms outside of the main focus of the research (all but the inference machinery, in this case) might be implemented in a simple, provisional way, but they would nevertheless be complete enough that the system could truly function by itself over a long period of time. If nothing at all could be done to build those other mechanisms in such a way that the system functioned at all, this would reflect badly on the logical reasoning mechanisms that are the focus of interest, but it would not invalidate them outright—that invalidation would only happen if, over the course of time, all attempts to improve the surrounding matrix of mechanisms ended in failure. (It is the feeling of this author that this eventual failure would probably happen, but the final arbiter of such a question would be empirical experiment).

More likely, frameworks would be used to explore a variety of non-traditional approaches to cognition, most of them in the generalized connectionist tradition. Whatever the philosophy that informs any given framework, however, the basic rules of the game would be to make both the framework and the systems instantiated within that framework complete enough to allow systematic variation of system parameters, and systematic comparison of the observed behavior of those systems.

This progression from general, loosely specified frameworks down to particular systems is a vital part of the process, and the framework part of the paradigm should not be dismissed as superfluous or scorned as unscientific. Anyone should be able to write down such a framework, for consideration by the community, so long as it is capable of being turned into a specific generator that yields instances of cognitive systems.

5. Conclusion

There is only space here to give a brief outline of the proposed theoretical psychology approach to AI, but even from this short account, it is clearly ambitious. One of its objectives is to persuade researchers to build and examine dozens of different types of system in a day, rather one type of system per career. Is this just a blue sky fantasy?

The way to make it possible is by means of a software development environment specifically designed to facilitate the building and testing of large numbers of similar, parameterized cognitive systems. The author is currently working on such a tool, the details of which will be covered in a later paper.

More generally, any framework that purports to be the basis for a particular approach to AI—whether that framework is within the theoretical psychology paradigm or not—should be defined and represented by a software development environment that allows systems in that class to be constructed with relative ease.

What is needed, in fact, is not so much a software development environment as an SDE generator that enables anyone to build a customized tool for building their chosen class of cognitive systems. Such a generator would support a variety of frameworks, not just one.

5.1. Scruffs versus Neats

One way to summarize the philosophy behind this paper is to say that AI research has gone though two phases, and is about to enter a third. In their influential textbook of AI, first published in 1995, Stuart Russell and Peter Norvig point to a change in attitude in the AI community that occurred in the mid-1980s:

> Recent years have seen a sea change in both the content and the methodology of research in artificial intelligence. It is now more common to build on existing theories than to propose brand new ones, to base claims on rigorous theorems or hard experimental evidence rather than on intuition, and to show relevance to real-world applications rather than toy examples. Some have characterized this change as a victory of the **neats** – those who think that AI theories should be grounded in mathematical rigor – over the **scruffies** – those who would rather try out lots of ideas, write some programs, and then assess what seems to be working. Both approaches are important. A shift toward increased neatness implies that the field has reached a level of stability and maturity. (Whether that stability will be disrupted by a new scruffy idea is another question). [14]

Put this way, the distinction between neats and scruffies reflects very favorably on those who dominate AI research today.

The present paper disputes this analysis: the Scruffy era was defined by an engineering attitude to the problem, while the present day Neat era is defined by a mathematical attitude that gives a comforting illusion of rigor simply because of the halo effect caused by large amounts of proof and deduction. The transition from

Scruffy to Neat looks more like the transition from Intuitionism to Behaviorism in psychology, and Neat AI has the same spurious aura of rigor that Behaviorism had.

No amount of mathematics can compensate for fundamental axioms that, when examined closely, turn out to be just as speculative as the ones that drove the engineers who came before. The complex systems perspective would argue that the overall performance of Neat AI systems will only be clear when complete examples of such systems—including all of the sensorimotor and learning mechanisms needed to ground them—are actually available for inspection and have been shown to converge on real intelligent behavior. Neat AI has scrupulously avoided such a showdown, so there is not yet any reason to believe that the assumptions at the root of the impressive-seeming mathematics are any more valid than the Scruffy assumptions that came before.

What we need right now is neither engineering nor mathematics, but science. We should be devising carefully controlled experiments to ask about the behavior of different kinds of systems, rather than exploring a few plausible systems chosen by instinct, or augmenting the same kind of instinctually-chosen systems with mathematics as a way to make them seem less arbitrary and more rigorous. Both of those old approaches involve assumptions about the relationship between the high-level functionality of AI systems and their low-level mechanisms which, from the point if view of the Complex Systems Problem, are untenable.

References

[1] Waldrop, M. M. (1992) "Complexity: The emerging science at the edge of order and chaos." Simon & Schuster, New York, NY.

[2] Holland, J. H. (1998) "Emergence." Helix Books, Reading, MA.

[3]Horgan, J. (1995) "From complexity to perplexity." Scientific American 272(6): 104-109.

[4] Wolfram, S. (2002) "A New Kind of Science." Wolfram Media: Champaign, IL. 737-750.

[5] Gardner, M. (1970) "Mathematical Games: The fantastic combinations of John Conway's new solitaire game 'life'." Scientific American 223(4): 120-123.

[6] Guy, R. K. (1985) "John Horton Conway," in Albers and G L Alexanderson (eds.), "Mathematical people: Profiles and interviews." Cambridge, MA: 43-50.

[7] Kohonen, T. (1987) "Self-organization and associative memory." Springer: Berlin.

[8] McClelland, J.L., Rumelhart, D.E. & Hinton, G.E. (1986) "The appeal of parallel distributed processing." In D.E. Rumelhart, J.L. McClelland & G.E. Hinton and the PDP Research Group, "Parallel distributed processing: Explorations in the microstructure of cognition, Volume 1." MIT Press: Cambridge, MA.

[9] Gorman R.P. and Sejnowski, T.J. (1988) "Analysis of hidden units in a layered network trained to classify sonar targets," Neural Networks, 1(1), 75–89.

[10] David E. Rumelhart, D.E., Hinton, G.E. and Williams, R.J. (1986) "Learning representations by back-propagating errors," Nature 323:533-536.

[11] Ackley, D.H., Hinton, G.E. and Sejnowski, T.J. (1985) "A learning algorithm for Boltzmann machines," Cognitive Science 9:147-169.

[12] Harley, T. A. (2004). "Promises, promises. Reply to commentators." Cognitive Neuropsychology, 21,51-56.

[13] Harley, T. A. (2004). "Does cognitive neuropsychology have a future?" Cognitive Neuropsychology, 21, 3-16.

[14] Russell, S. J. and Norvig, P. (1995) "Artificial Intelligence: A modern approach." Prentice Hall, Upper Saddle River, NJ.

[15] Kuhn, T.S. (1962) "The structure of scientific revolutions." University of Chicago Press, Chicago, IL.

B. Goertzel and P. Wang (Eds.)
IOS Press, 2007
© 2007 The authors and IOS Press. All rights reserved.

Stages of Cognitive Development in Uncertain-Logic-Based AI Systems

Ben GOERTZEL[a], Stephan Vladimir BUGAJ[b]

[a]*Novamente, LLC and Virginia Tech,*
[b]*AGI Research Institute*
ben@goertzel.org, stephan@bugaj.com

Abstract. A novel theory of stages in cognitive development is presented, loosely corresponding to Piagetan theory but specifically oriented toward AI systems centered on uncertain inference components. Four stages are articulated (infantile, concrete, formal and reflexive), and are characterized both in terms of external cognitive achievements (a la Piaget) and in terms of internal inference control dynamics. The theory is illustrated via the analysis of specific problem solving tasks corresponding to the different stages. The Novamente AI Engine, with its Probabilistic Logic Networks uncertain inference component and its embodiment n the AGI-SIM simulation world, is used as an example throughout.

Introduction

Contemporary cognitive science contains essentially no theory of "AI developmental psychology" – a lack which is frustrating from the perspective of AI scientists concerned with understanding, designing and controlling the cognitive development of generally intelligent AI systems. There is of course an extensive science of human developmental psychology, and so it is a natural research program to take the chief ideas from the former and inasmuch as possible port them to the AI domain. However this is not an entirely simple matter both because of the differences between humans and AI's and because of the unsettled nature of contemporary developmental psychology theory. The present paper describes some work that we have done in this direction, as part of a longer-term project to develop a systematic theory of AI cognitive development.

The ghost of Jean Piaget hangs over modern developmental psychology in a yet unresolved way. Piaget's theories provide a cogent overarching perspective on human cognitive development, coordinating broad theoretical ideas and diverse experimental results into a unified whole. Modern experimental work has shown Piaget's ideas to be often oversimplified and incorrect. However, what has replaced the Piagetan understanding is not an alternative unified and coherent theory, but a variety of microtheories addressing particular aspects of cognitive development. For this reason a number of contemporary theorists taking a computer science [1] or dynamical systems [2-4] approach to developmental psychology have chosen to adopt the Piagetan framework in spite of its demonstrated shortcomings, both because of its conceptual strengths and for lack of a coherent, more rigorously grounded alternative.

Table 1

Stage	Example
Infantile	Object Permanence
Concrete	Conservation of Number, Theory of Mind
Formal	Systematic Experimentation
Reflexive	Correction of Inference Bias

The work described here involves the construction of a theory of cognitive development inspired conceptually by Piaget's work, but specifically applicable to AI systems that rely on uncertain logical inference as a primary or highly significant component. Piaget describes a series of stages of cognitive development, each corresponding to a certain level of sophistication in terms of the types of reasoning a child can carry out. We describe a related series of stages, each corresponding not only to a level of sophistication in terms of demonstrated problem-solving ability, but also to a level of internal sophistication in terms of inference control mechanisms within AI software implementations.

This work was inspired by our ongoing research involving the Novamente AI Engine [5-7], a complex integrative software system aimed at achieving advanced Artificial General Intelligence (AGI) [8]. The Novamente system has been integrated with AGI-SIM, a 3D simulation world powered by the CrystalSpace game engine used in the Crystal Cassie embodiment of the SNePs AGI system [9]. Table 1 above shows each of our proposed developmental stages, with examples drawn from our ongoing research with the Novamente system.

1.Piaget's Approach to Cognitive Development

Jean Piaget, in his classic studies of human developmental psychology [10-15], conceived of child development in four stages, each roughly identified with an age group: infantile, preoperational, concrete operational, and formal.

--*Infantile*: In this stage a mind develops basic world-exploration driven by instinctive actions. Reward-driven reinforcement of actions learned by imitation, simple associations between words and objects, actions and images, and the basic notions of time, space, and causality are developed. The most simple, practical ideas and strategies for action are learned.

--*Preoperational*: At this stage we see the formation of mental representations, mostly poorly organized and un-abstracted, building mainly on intuitive rather than logical thinking. Word-object and image-object associations become systematic rather than occasional. Simple syntax is mastered, including an understanding of subject-argument relationships. One of the crucial learning achievements here is "object permanence”--infants learn that objects persist even when not observed. However, a number of cognitive failings persist with respect to reasoning about logical operations, and abstracting the effects of intuitive actions to an abstract theory of operations.

--*Concrete*: More abstract logical thought is applied to the physical world at this stage. Among the feats achieved here are: reversibility--the ability to undo steps already done; conservation--understanding that properties can persist in spite of appearances;

theory of mind--an understanding of the distinction between what I know and what others know (If I cover my eyes, can you still see me?). Complex concrete operations, such as putting items in height order, are easily achievable. Classification becomes more sophisticated, yet the mind still cannot master purely logical operations based on abstract logical representations of the observational world.

--*Formal*: Abstract deductive reasoning, the process of forming, then testing hypotheses, and systematically reevaluating and refining solutions, develops at this stage, as does the ability to reason about purely abstract concepts without reference to concrete physical objects. This is adult human-level intelligence. Note that the capability for formal operations is intrinsic in the PTL component of Novamente, but in-principle capability is not the same as pragmatic, grounded, controllable capability.

Despite the influence and power of Piaget’s theory, it has received much valid criticism. Very early on, Vygotsky [16, 17] disagreed with Piaget’s explanation of his stages as inherent and developed by the child’s own activities, and Piaget’s prescription of good parenting as not interfering with a child’s unfettered exploration of the world. Much of the analysis of Piaget’s stages as being asocially grounded start with Vygotsky’s assertion that children function in a world surrounded by adults who provide a cultural context, offering ongoing assistance, critique, and ultimately validation of the child’s developmental activities.

Vygotsky also was an early critic with respect to the idea that cognitive development is continuous, and continues beyond Piaget’s formal stage. Gagne [18] also believes in continuity, and that learning of prerequisite skills made the learning of subsequent skills easier and faster without regard to Piagetan stage formalisms. Subsequent researchers have argued that Piaget has merely constructed ad hoc descriptions of the sequential development of behaviour [19-22]. We agree that learning is a continuous process, and our notion of stages is more statistically constructed than rigidly quantized.

Critique of Piaget’s notion of transitional “half stages” is also relevant to a more comprehensive hierarchical view of development. Some have proposed that Piaget’s half stages are actually stages [23]. As Commons and Pekker [22] point out: “the definition of a stage that was being used by Piaget was based on analyzing behaviors and attempting to impose different structures on them. There is no underlying logical or mathematical definition to help in this process…” Their Hierarchical Complexity development model uses task achievement rather than ad hoc stage definition as the basis for constructing relationships between phases of developmental ability--an approach which we find useful, though our approach is different in that we define stages in terms of specific underlying cognitive mechanisms.

Another critique of Piaget is that one individual’s performance is often at different ability stages depending on the specific task (for example [24]). Piaget responded to early critiques along these lines by calling the phenomenon “horizontal décalage,” but neither he nor his successors [25,26] have modified his theory to explain (rather than merely describe) it. Similarly to Thelen and Smith [2], we observe that the abilities encapsulated in the definition of a certain stage emerge gradually during the previous stage--so that the onset of a given stage represents the mastery of a cognitive skill that was previously present only in certain contexts.

Piaget also had difficulty accepting the idea of a preheuristic stage, early in the infantile period, in which simple trial-and-error learning occurs without significant heuristic guidance [27], a stage which we suspect exists and allows formulation of heuristics by aggregation of learning from preheuristic pattern mining. Coupled with his belief that a mind's innate abilities at birth are extremely limited, there is a troublingly unexplained transition from inability to ability in his model.

Finally, another limiting aspect of Piaget's model is that it did not recognize any stages beyond formal operations, and included no provisions for exploring this possibility. A number of researchers [25,28-31] have described one or more postformal stages. Commons and colleagues have also proposed a task-based model which provides a framework for explaining stage discrepancies across tasks and for generating new stages based on classification of observed logical behaviors. [32] promotes a statistical conception of stage, which provides a good bridge between task-based and stage-based models of development, as statistical modeling allows for stages to be roughly defined and analyzed based on collections of task behaviors.

[29] postulates the existence of a postformal stage by observing *elevated levels of abstraction* which, they argue, are not manifested in formal thought. [33] observes a postformal stage when subjects become capable of analyzing and coordinating complex logical systems with each other, creating metatheoretical supersystems. In our model, with the reflexive stage of development, we expand this definition of metasystemic thinking to include the ability to consciously refine one's own mental states and formalisms of thinking. Such self-reflexive refinement is necessary for learning which would allow a mind to analytically devise entirely new structures and methodologies for both formal and postformal thinking.

2.The Uncertain Inference Paradigm

Piaget's developmental stages are very general, referring to overall types of learning, not specific mechanisms or methods. This focus was natural since the context of his work was *human* developmental psychology, and neuroscience has not yet progressed to the point of understanding the neural mechanisms underlying any sort of inference. But if one is studying developmental psychology in an AI context where one knows something about the internal mechanisms of the AI system under consideration, then one can work with a more specific model of learning. Our focus here is on AI systems whose operations contain uncertain inference as a central component, both directly and used as a model for a theory which we hope to eventually test against natural intelligence as well.

An uncertain inference system, as we consider it here, consists of four components, which work together in a feedback-control loop (Fig. 1):

- a content representation scheme
- an uncertainty representation scheme
- a set of inference rules
- a set of inference control schemata

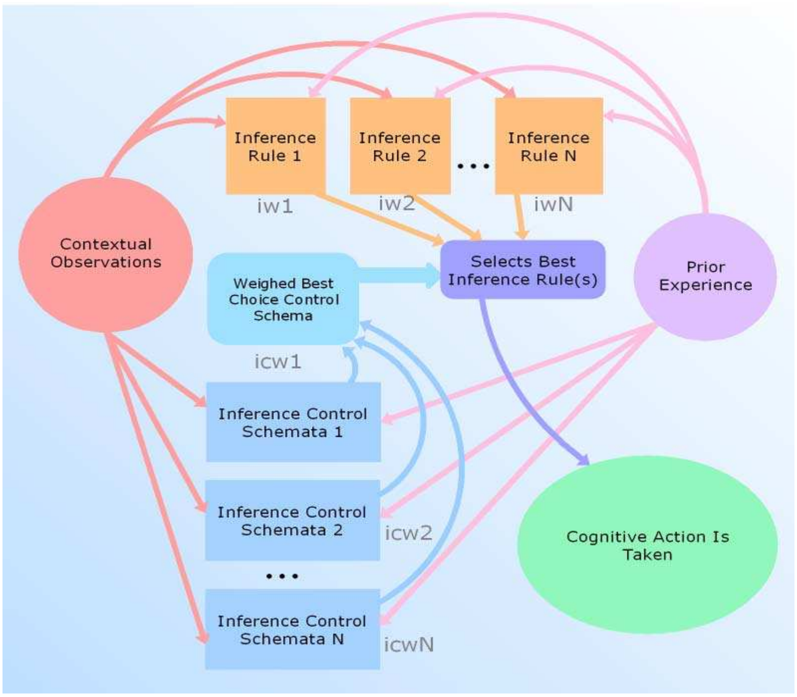

Inference Rule 1
Inference Rule 2
Inference Rule N
iw1
iw2
iwN
Contextual Observations
Weighed Best Choice Control Schema
Selects Best Inference Rule(s)
Prior Experience
icw1
Inference Control Schemata 1
Inference Control Schemata 2
icw2
Inference Control Schemata N
icwN
Cognitive Action Is Taken

Figure 1. A Simplified Look at Feedback-Control in Uncertain Inference

Examples of content representation schemes are predicate logic and term logic [34]. Examples of uncertainty representation schemes are fuzzy logic [35], imprecise probability theory [36,37], Dempster-Shafer theory [37,38], Bayesian probability theory [39], NARS [40], and the Probabilistic Logic Networks (PLN) representation used in Novamente [41].

Many, but not all, approaches to uncertain inference involve only a limited, weak set of inference rules (e.g. not dealing with complex quantified expressions). Both NARS and PLN contain uncertain inference rules that apply to logical constructs of arbitrary complexity. Only a system capable of dealing with arbitrary complexity will have any potential of leading to real intelligence.

The subtlest part of uncertain inference is inference control: the choice of which inferences to do, in what order. Inference control is the primary area in which human inference currently exceeds automated inference. Humans are not very efficient or accurate at carrying out inference rules, with or without uncertainty, but we are very good at determining which inferences to do and in what order, in any given context. The lack of effective, context-sensitive inference control heuristics is why the general ability of current automated theorem provers is considerably weaker than that of a mediocre university mathematics major [42].

3.Novamente and Probabilistic Logic Networks

Novamente's knowledge representation consists of weighted labeled, generalized hypergraphs. Patterns embodying knowledge emerge from applying various learning and reasoning algorithms to these hypergraphs.

A hypergraph is an abstract mathematical structure, which consists of objects called Vertices and objects called Edges, which connect the Vertices [43]. In Novamente we have adopted the terminology of using *Node/Vertex* to refer to the elements of the hypergraph that are concretely implemented in a Novamente system's memory, and *Link/Edge* to refer to elements of hypergraphs that are used to model Novamente systems and represent patterns that emerge in the concretely implemented hypergraph. We use the term *Atom* to refer to Nodes and Links inclusively. A hypergraph differs from a graph in that it allows Edges to connect more than two Vertices. Novamente hypergraphs extend ordinary hypergraphs to contain additional features, such as Edges that point to Edges instead of Vertices, and Vertices that represent complete sub-hypergraphs.

A "weighted, labeled hypergraph" is a hypergraph whose Atoms all have associated annotations called *labels*, and one or more numbers that are generically called *weights*. The label associated with an Atom might be interpreted as telling you what *type* of entity it is (a metalogical knowledge annotation). An example of a weight attached to an Atom is a number representing a probability, or a number representing how important the Atom is to the system.

In the framework introduced in the previous section, Novamente's content representation is a "labeled generalized hypergraph with weights representing the attention paid to hypergraph components via learning and reasoning algorithms" and the uncertainty representation consists of some additional weights attached to the Nodes and Links of the hypergraph, representing probability values and related quantities such as "weight of evidence."

Novamente's knowledge representation includes various types of Nodes, including ConceptNodes and SchemaNodes. SchemaNodes embody cognitive, perceptual or motoric procedures, and are represented as mathematical objects using arithmetic, logical and combinatory operators to combine elementary data types and Novamente Nodes and Links. It also includes a number of other node types including PredicateNodes (SchemaNodes that produce truth values as their outputs) and Nodes representing particular kinds of concrete information, such as NumberNodes, WordNodes, PolygonNodes, etc. An extensive list is given in [6].

Novamente also contains a variety of Link types, including some that represent logical relationships, such as ExtentionalInheritanceLink (ExtInhLink: an edge which indicates that the source Atom is a special case of the target), ExtensionalSimilarityLink (ExtSimLink: which indicates that one Atom is similar to another), and ExecutionLink (a ternary edge, which joins {S,B,C} when S is a SchemaNode and the result from applying S to B is C). Thus, a Novamente knowledge network is a hypergraph whose Nodes represent ideas or procedures, and whose Links represent relationships of specialization, similarity or transformation among ideas and/or procedures.

ExtInh and ExtSim Links come with probabilistic weights indicating the extent of the relationship they denote (e.g. the ExtSimLink joining the "cat" ConceptNode to the "dog" ConceptNode gets a higher probability weight than the one joining the "cat" ConceptNode to the "washing machine" ConceptNode). The mathematics of transformations involving these probabilistic weights becomes quite involved--particularly when one introduces SchemaNodes corresponding to abstract mathematical operations. SchemaNodes enable Novamente hypergraphs to have the complete mathematical power of standard logical formalisms like predicate calculus, but with the added advantage of a natural representation of uncertainty in terms of probabilities, as well as a neurostructurally motivated model of complex knowledge as dynamical networks.

Novamente contains a probabilistic reasoning engine called Probabilistic Logic Networks (PLN) which exists specifically to carry out reasoning on these relationships, and will be described in a forthcoming publication [8]. The mathematics of PLN contains many subtleties, and there are relations to prior approaches to uncertain inference including NARS [40] and Walley's theory of interval probabilities [44]. The current implementation of PLN within the Novamente software has been tested on various examples of mathematical and commonsense inference.

A simple example of a PLN uncertain inference rule is the probabilistic deduction rule, which takes the form

A ➔ B
B ➔ C
|-
A ➔ C

(where e.g. A➔B is a shorthand for the ExtInhLink from A to B), whose uncertain truth value formula has as one component the formula

$$\mathbf{s_{AC} = s_{AB}\, s_{BC} + (1-s_{AB})\,(s_C - s_B\, s_{BC}) / (1- s_B)}$$

(where e.g. s_{AC} and s_B refer to the probability values attached to A➔C and B respectively). PLN attaches to each node and link a "weight of evidence" value in addition to a probability, but the deduction formula for weight of evidence is more complex and will not be given here.

Inference control in Novamente takes several forms:

1. Standard forward-chaining and backward-chaining inference heuristics (see e.g. [45])
2. A reinforcement learning mechanism that allows inference rules to be chosen based on experience. Probabilities are tabulated regarding which inference rules have been useful in the past in which contexts, and these are subsequently used to bias the choices of inference rules during forward or backward chaining inference
3. Application of PLN inference to the probabilities used in the reinforcement learning mechanism--enables generalization, abstraction and analogy to be used in guessing which inference rules are most useful in a given context

These different approaches to inference control enable increasingly complex inferences, and involve increasing amounts of processor-time utilization and overall cognitive complexity. They may also be interpreted as corresponding to loosely Piagetan stages of cognitive development.

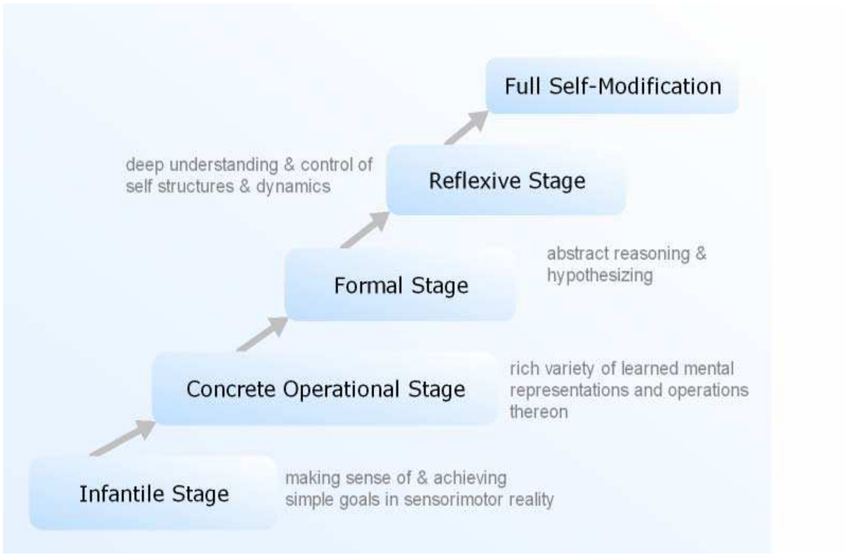

Full Self-Modification
deep understanding & control of self structures & dynamics
Reflexive Stage
abstract reasoning & hypothesizing
Formal Stage
Concrete Operational Stage
rich variety of learned mental representations and operations thereon
Infantile Stage
making sense of & achieving simple goals in sensorimotor reality

Figure 2. The Stages of the Goertzel-Bugaj Theory

4.Defining Developmental Stages in Terms of Inference Control

Inspired by Piaget's general ideas, later critiques, and the structure of inference control in Novamente, we have created a novel theory of cognitive developmental stages (Fig. 2), defined in terms of the control of uncertain inference trajectories.

Each stage in our theory is defined in terms of both testable cognitive capabilities, similar to Piaget and other researchers in the field, but also in terms of inference control structures which we feel serve as both a reasonable model for natural intelligence and also are suitable for application within an AI system.

Inference control structures are mechanisms by which the process of learning itself is performed. Ability to learn is dictated by the capabilities of the inference control system, and these capabilities are refined through iterative experience and observational feedback just as they use experience and observation to refine capabilities in other cognitive tasks.

By defining these stages in terms of inference control, we have an structural argument to make about the topological shape and complexity of the underlying cognitive network in addition to the traditional capability-based arguments about what defines an intelligent entity as being in a particular stage (for a particular task and its associated cognitive pathways). This is applicable both to building AI systems and in providing structural hypotheses about natural intelligence which can be tested as high-resolution, continuous-time neural imaging technologies mature and allow us to do so.

So, while our stages draw upon Piaget and other research, it is a focus on tying underlying learning procedure as structure with capability that is one of the differences in our theory. The stages are defined as follows.

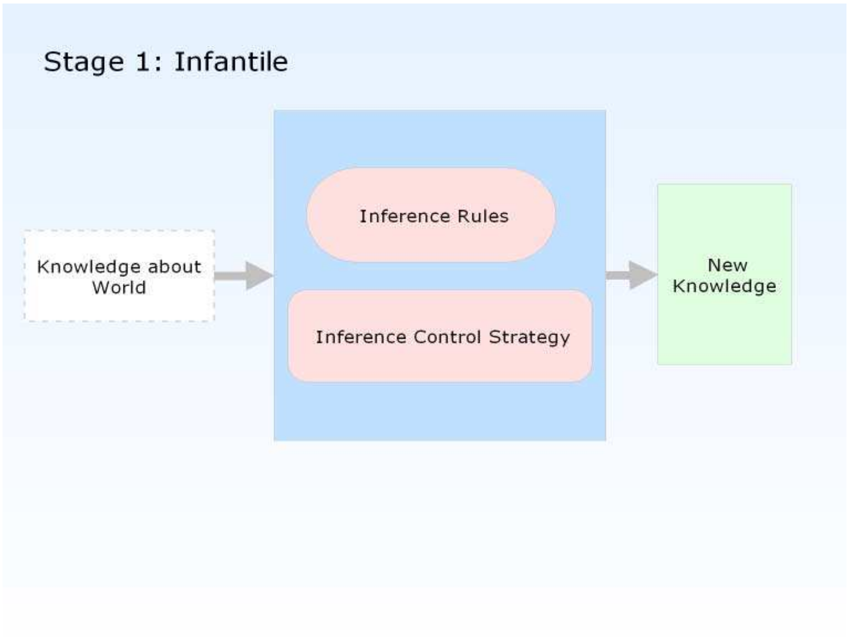


Figure 3. The Infantile Stage

--*Infantile*: Able to recognize patterns in and conduct inferences about the world, but only using simplistic hard-wired (not experientially learned) inference control schema, along with pre-heuristic pattern mining of experiential data.

In the infantile stage an entity is able to recognize patterns in and conduct inferences about its sensory surround context (i.e., it's "world"), but only using simplistic, hard-wired (not experientially learned) inference control schemata. Preheuristic pattern-mining of experiential data is performed in order to build future heuristics about analysis of and interaction with the world.

Infantile stage tasks include:

- Exploratory behavior in which useful and useless / dangerous behavior is differentiated by both trial and error observation, and by parental guidance.
- Development of "habits" -- i.e. Repeating tasks which were successful once to determine if they always / usually are so.
- Simple goal-oriented behavior such as "find out what cat hair tastes like" in which one must plan and take several sequentially dependent steps in order to achieve the goal.

Inference control is very simple during the infantile stage (Fig. 3) as it is the stage during which both the most basic knowledge of the world is acquired, and the most basic of cognition and inference control structures are developed as the building block upon which will be built the next stages of both knowledge and inference control.

Another example of a cognitive task at the borderline between infantile and concrete cognition is learning object permanence, a problem discussed in a Novamente/AGI-SIM context in [46]. Another example is the learning of word-object associations: e.g. learning that when the word "ball" is uttered in various contexts ("Get me the ball," "That's a nice ball," etc.) it generally refers to a certain type of object. The key point regarding these "infantile" inference problems, from the Novamente perspective, is that assuming one provides the inference system with an appropriate set of perceptual and motor ConceptNodes and SchemaNodes, the chains of inference involved are short. They involve about a dozen inferences, and this means that the search tree of possible PLN inference rules walked by the PLN backward-chainer is relatively shallow. Sophisticated inference control is not required: standard AI heuristics are sufficient.

In short, textbook narrow-AI reasoning methods, utilized with appropriate uncertainty-savvy truth value formulas and coupled with appropriate representations of perceptual and motor inputs and outputs, correspond roughly to Piaget's infantile stage of cognition. The simplistic approach of these narrow-AI methods may be viewed as a method of creating building blocks for subsequent, more sophisticated heuristics.

In our theory Piaget's preoperational phase appears as transitional between the infantile and concrete operational phases. We suspect this approach to cognitive modeling may have general value beyond Novamente, but we will address a more generalized developmental theory in future writings. We have designed specific Novamente / AGI-SIM learning tasks based on all the key Piagetan themes. Currently our concrete work is near the beginning of this list, at Piaget's infantile stage.

--*Concrete:* Able to carry out more complex chains of reasoning regarding the world, via using inference control schemata that adapt behavior based on experience (reasoning about a given case in a manner similar to prior cases).

In the concrete operational stage an entity is able to carry out more complex chains of reasoning about the world. Inference control schemata which adapt behavior based

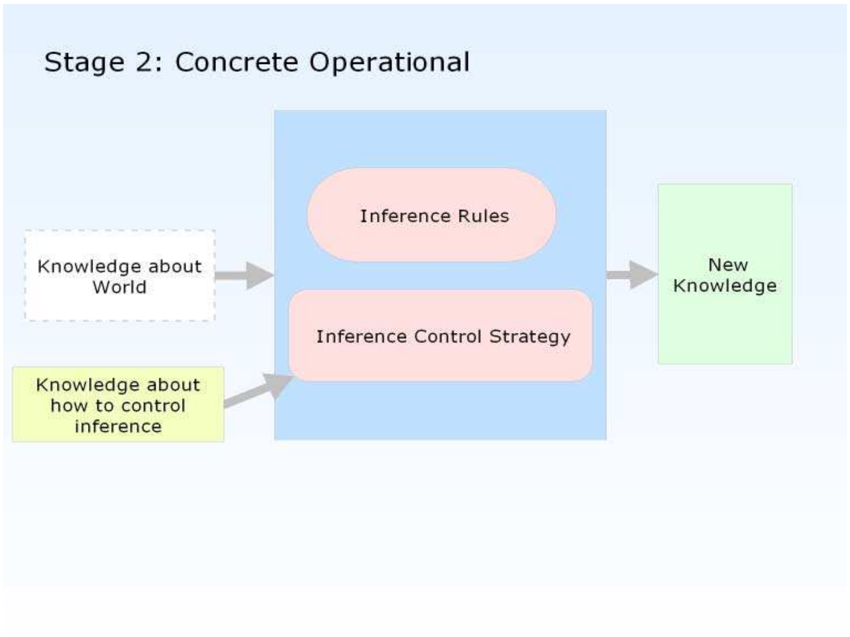

Stage 2: Concrete Operational
Inference Rules
Knowledge about World
Inference Control Strategy
Knowledge about how to control inference
New Knowledge

Figure 4. The Concrete Operational Stage

on experience, using experientially learned heuristics (including those learned in the prior stage), are applied to both analysis of and interaction with the sensory surround / world.

At this stage a special cognitive task capability is gained. It is referred to as "Theory of Mind." In cognitive science "Theory of Mind" means the ability to understand the fact that not only oneself, but other sentient beings have memories, perceptions, and experiences. This is the ability to conceptually "put oneself in another's shoes" (even if you happen to assume incorrectly about them by doing so).

Concrete Operational stage tasks include:

- Conservation tasks, such as conservation of number,
- Decomposition of complex tasks into easier subtasks, allowing increasingly complex tasks to be approached by association with more easily understood (and previously experienced) smaller tasks,
- Classification and Serialization tasks, in which the mind can cognitively distinguish various disambiguation criteria and group or order objects accordingly.

.In terms of inference control this is the stage in which actual knowledge about how to control inference itself is first explored (Fig. 4). This means an emerging understanding of inference itself as a cognitive task and methods for learning, which will be further developed in the following stages.

--*Formal*: Able to carry out arbitrarily complex inferences (constrained only by computational resources) via including inference control as an explicit subject of abstract learning.

In the formal stage, an entity is able to carry out arbitrarily complex inferences (constrained only by computational resources). Abstraction and inference about both

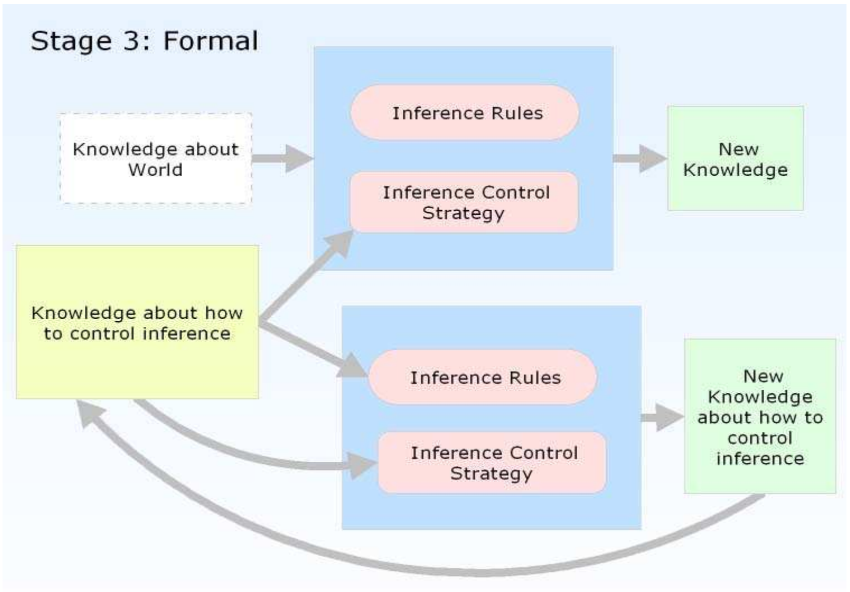

Stage 3: Formal
Knowledge about World
Inference Rules
Inference Control Strategy
New Knowledge
Knowledge about how to control inference
Inference Rules
Inference Control Strategy
New Knowledge about how to control inference

Figure 5. The Formal Stage

the sensorimotor surround (world) and about abstract ideals themselves (including the final stages of indirect learning about inference itself) are fully developed.

Formal stage tasks are centered entirely around abstraction and higher-order inference tasks such as:

- Mathematics and other formalizations.
- Scientific experimentation and other rigorous observational testing of abstract formalizations.
- Social and philosophical modeling, and other advanced applications of empathy and the Theory of Mind.

In terms of inference control this stage sees not just perception of new knowledge about inference control itself, but inference controlled reasoning about that knowledge and the creation of abstract formalizations about inference control which are reasoned-upon, tested, and verified or debunked (Fig.5).

Existing natural intelligence systems (i.e., humans) are fully capable of performing up to the Formal stage.

It is more controversial whether or not any humans have truly mastered the following stage, the reflexive stage. Followers of various meditative and pedagogical practices claim Reflexive stage abilities, but such claims are not as yet considered verified.

--*Reflexive*: Capable of self-modification of internal structures. (In the case of a Novamente, this process is very direct and thorough.)

In the reflexive stage an entity is able to include inference control itself as an explicit subject of abstract learning (i.e. the ability to reason about one's own tactical

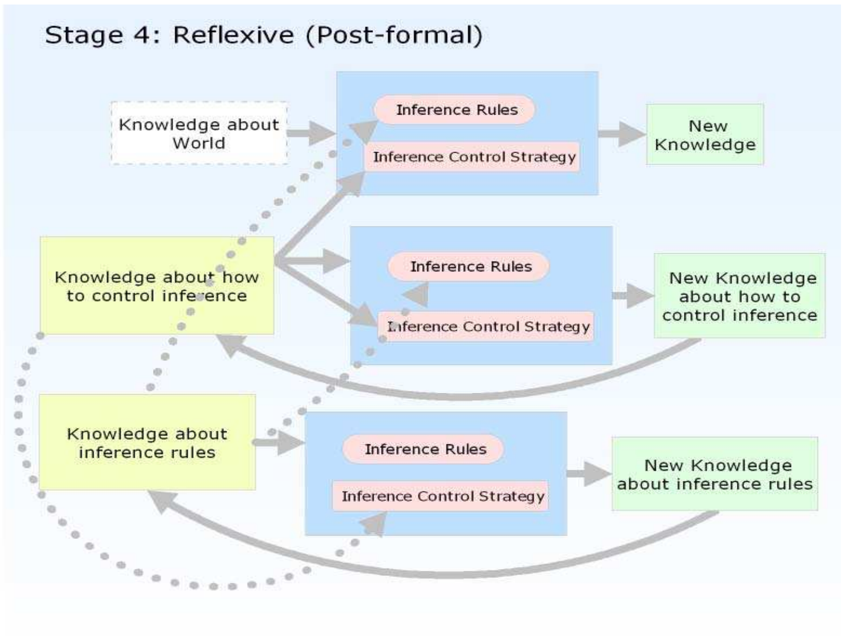

Stage 4: Reflexive (Post-formal)
Knowledge about World
Inference Rules
Inference Control Strategy
New Knowledge
Knowledge about how to control inference
Inference Rules
Inference Control Strategy
New Knowledge about how to control inference
Knowledge about inference rules
Inference Rules
Inference Control Strategy
New Knowledge about inference rules

Figure 6. The Reflexive (Post-formal) Stage and strategic approach to modifying one's own learning and thinking, and modify the these inference control strategies based on analysis of experience with various cognitive approaches.

Ultimately, the entity can self-modify its internal cognitive structures. Any knowledge or heuristics can be revised, including metatheoretical and metasystemic thought itself. Initially this is done indirectly, but at least in the case of AI systems it is theoretically possible to also do so directly. This is referred to as a separate stage of Full Self Modification in (Fig. 2) but it is really the end phase of the reflexive stage. Self modification of inference control itself is the primary task in this stage. In terms of inference control this stage adds an entire new feedback loop for reasoning about inference control itself (Fig. 6).

The semantics of our stages is similar but not identical to Piaget's. Our stages are defined via internal cognitive mechanisms, and we then posit that these mechanisms correspond to the general ability to solve certain classes of problems in a generalizable way. For instance, we suggest that it is only through inference control schemata which adapt based on experience that uncertain inference-based AI systems can learn to consistently solve Piagetan concrete-operational tasks in a way that provides knowledge suitable for further generalization. However, it may be that minds using hard-wired inference control schemata (typical of the infantile stage) can still solve some Piagetan concrete-operational tasks, though most solutions to such tasks obtained in this way will be "brittle" and not easily generalizable to other tasks using infantile cognition.

5. Conservation of Number Detailed

Above, we mentioned the idea of conservation of number. This is an example of a learning problem classically categorized within Piaget's concrete-operational phase, a "conservation laws" problem, discussed in [1] in the context of software that solves the problem using (logic-based and neural net) narrow-AI techniques. Conservation laws are very important to cognitive development.

Conservation is the idea that a quantity remains the same despite changes in appearance. If you show a child some objects (Fig. 1) and then spread them out, an infantile mind will focus on the spread, and believe that there are now more objects than before, whereas a concrete-operational mind will understand that the quantity of objects has not changed.

Conservation of number seems very simple, but from a developmental perspective it is actually rather difficult. "Solutions" like those given in [1] that use neural networks or customized logical rule-bases to find specialized solutions that solve only this problem fail to fully address the issue, because these solutions don't create knowledge adequate to aid with the solution of related sorts of problems.

We hypothesize that this problem is hard enough that for an inference-based AI system to solve it in a developmentally useful way, its inferences must be guided by meta-inferential lessons learned from prior similar problems. When approaching a number conservation problem, for example, a reasoning system might draw upon past experience with set-size problems (which may be trial-and-error experience). This is not a simple "machine learning" approach whose scope is restricted to the current problem, but rather a heuristically guided approach which (a) aggregates information from prior experience to guide solution formulation for the problem at hand, and (b) adds the present experience to the set of relevant information about quantification problems for future refinement of thinking.

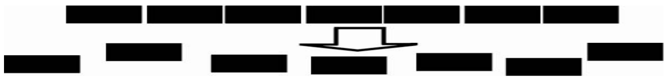


Figure 7. Conservation of Number

For instance, a very simple context-specific heuristic that a system might learn would be: “When evaluating the truth value of a statement related to the number of objects in a set, it is generally not that useful to explore branches of the backwards-chaining search tree that contain relationships regarding the sizes, masses, or other physical properties of the objects in the set.” This heuristic itself may go a long way toward guiding an inference process toward a correct solution to the problem--but it is not something that a mind needs to know “a priori.” A concrete-operational stage mind may learn this by data-mining prior instances of inferences involving sizes of sets. Without such experience-based heuristics, the search tree for such a problem will likely be unacceptably large. Even if it is “solvable” without such heuristics, the solutions found may be overly fit to the particular problem and not usefully generalizable.

6. Theory of Mind Detailed

Another, absolutely crucial, learning problem mentioned above that is typically classed in the Piagetan concrete-operational stage is ”theory of mind” – which means, in this context, fully understanding the fact that others have memories, perceptions and experiences.

Consider this experiment: a preoperational child is shown her favorite “Dora the Explorer” DVD box. Asked what show she’s about to see, she’ll answer “Dora.” However, when her parent plays the disc, it’s “Spongebob Squarepants.” If you then ask her what show her friend will expect when given the “Dora” DVD box, she will respond “Spongebob” although she just answered “Dora” for herself. A child lacking a theory of mind can not reason through what someone else would think given knowledge other than her own current knowledge. Knowledge of self is intrinsically related to the ability to differentiate oneself from others, and this ability may not be fully developed at birth.

Several theorists [47,48], based in part on experimental work with autistic children, perceive theory of mind as embodied in an innate module of the mind activated at a certain developmental stage (or not, if damaged). While we consider this possible, we caution against adopting a simplistic view of the “innate vs. acquired” dichotomy: if there is innateness it may take the form of an innate predisposition to certain sorts of learning [49].

Davidson [50], Dennett [51] and others support the common belief that theory of mind is dependent upon linguistic ability. A major challenge to this prevailing philosophical stance came from Premack and Woodruff [49] who postulated that prelinguistic primates do indeed exhibit “theory of mind” behavior. While Premack and Woodruff’s experiment itself has been challenged [52], their general result has been bolstered by follow-up work showing similar results such as [53]. It seems to us that while theory of mind depends on many of the same inferential capabilities as language learning, it is not intrinsically dependent on the latter.

There is a school of thought often called the *Theory Theory* [54]-[55]-[56] holding that a child's understanding of mind is best understood in terms of the process of iteratively formulating and refuting a series of naïve theories about others. Alternately, Gordon [57] postulates that theory of mind is related to the ability to run cognitive simulations of others' minds using one's own mind as a model. We suggest that these two approaches are actually quite harmonious with one another. In an uncertain AI context, both theories and simulations are grounded in collections of uncertain implications, which may be assembled in context-appropriate ways to form theoretical conclusions or to drive simulations. Even if there is a special "mind-simulator" dynamic in the human brain that carries out simulations of other minds in a manner fundamentally different from explicit inferential theorizing, the inputs to and the behavior of this simulator may take inferential form, so that the simulator is in essence a way of efficiently and implicitly producing uncertain inferential conclusions from uncertain premises.

The details via which a Novamente system should be able to develop theory of mind in the AGI-SIM world have been articulated in detail, though practical learning experiments in this direction have not yet been done. We have not yet explored the possibility of giving Novamente a special "mind-simulator" component, though this would be possible; instead we have initially been pursuing a more purely inferential approach.

First, it is very simple for a Novamente system to learn patterns such as "If I rotated by pi radians, I would see the yellow block." And it's not a big leap for PLN to go from this to the recognition that "You look like me, and you're rotated by pi radians relative to my orientation, therefore you probably see the yellow block." The only nontrivial aspect here is the "you look like me" premise.

Recognizing "embodied agent" as a category, however, is a problem fairly similar to recognizing "block" or "insect" or "daisy" as a category. Since the Novamente agent can perceive most parts of its own "robot" body--its arms, its legs, etc.--it should be easy for the agent to figure out that physical objects like these look different depending upon its distance from them and its angle of observation. From this it should not be that difficult for the agent to understand that it is naturally grouped together with other embodied agents (like its teacher), not with blocks or bugs.

The only other major ingredient needed to enable theory of mind is "reflection"--the ability of the system to explicitly recognize the existence of knowledge in its own mind (note that this term "reflection" is not the same as our proposed "reflexive" stage of cognitive development). This exists automatically in Novamente, via the built-in vocabulary of elementary procedures supplied for use within SchemaNodes (specifically, the atTime and TruthValue operators). Observing that "at time T, the weight of evidence of the link L increased from zero" is basically equivalent to observing that the link L was created at time T.

Then, the system may reason, for example, as follows (using a combination of several PLN rules including the above-given deduction rule):

Implication
    My eye is facing a block and it is not dark
    A relationship is created describing the block's color

```
Similarity
        My body
        My teacher's body
|-
Implication
        My teacher's eye is facing a block and it is not dark
        A relationship is created describing the block's color
```

This sort of inference is the essence of Piagetan “theory of mind.” Note that in both of these implications the created relationship is represented as a variable rather than a specific relationship. The cognitive leap is that in the latter case the relationship actually exists in the teacher’s implicitly hypothesized mind, rather than in Novamente’s mind. No explicit hypothesis or model of the teacher’s mind need be created in order to form this implication--the hypothesis is created implicitly via inferential abstraction. Yet, a collection of implications of this nature may be used via an uncertain reasoning system like PLN to create theories and simulations suitable to guide complex inferences about other minds.

From the perspective of developmental stages, the key point here is that in a Novamente context this sort of inference is too complex to be viably carried out via simple inference heuristics. This particular example must be done via forward chaining, since the big leap is to actually think of forming the implication that concludes inference. But there are simply too many combinations of relationships involving Novamente’s eye, body, and so forth for the PLN component to viably explore all of them via standard forward-chaining heuristics. Experience-guided heuristics are needed, such as the heuristic that if physical objects A and B are generally physically and functionally similar, and there is a relationship involving some part of A and some physical object R, it may be useful to look for similar relationships involving an analogous part of B and objects similar to R. This kind of heuristic may be learned by experience--and the masterful deployment of such heuristics to guide inference is what we hypothesize to characterize the concrete stage of development. The “concreteness” comes from the fact that inference control is guided by analogies to prior similar situations.

7. Systematic Experimentation

The Piagetan formal phase is a particularly subtle one from the perspective of uncertain inference. In a sense, AI inference engines already have strong capability for formal reasoning built in. Ironically, however, no existing inference engine is capable of deploying its reasoning rules in a powerfully effective way, and this is because of the lack of inference control heuristics adequate for controlling abstract formal reasoning. These heuristics are what arise during Piaget’s formal stage, and we propose that in the content of uncertain inference systems, they involve the application of inference itself to the problem of refining inference control.

A problem commonly used to illustrate the difference between the Piagetan concrete operational and formal stages is that of figuring out the rules for making pendulums swing quickly versus slowly [10]. If you ask a child in the formal stage to solve this problem, she may proceed to do a number of experiments, e.g. build a long string with a light weight, a long string with a heavy weight, a short string with a light weight and a short string with a heavy weight. Through these experiments she may determine that a short string leads to a fast swing, a long string leads to a slow swing, and the weight doesn't matter at all.

The role of experiments like this, which test "extreme cases," is to make cognition easier. The formal-stage mind tries to map a concrete situation onto a maximally simple and manipulable set of abstract propositions, and then reason based on these. Doing this, however, requires an automated and instinctive understanding of the reasoning process itself. The above-described experiments are good ones for solving the pendulum problem because they provide data that is very easy to reason about. From the perspective of uncertain inference systems, this is the key characteristic of the formal stage: formal cognition approaches problems in a way explicitly calculated to yield tractable inferences.

Note that this is quite different from saying that formal cognition involves abstractions and advanced logic. In an uncertain logic-based AI system, even infantile cognition may involve these--the difference lies in the level of inference control, which in the infantile stage is simplistic and hard-wired, but in the formal stage is based on an understanding of what sorts of inputs lead to tractable inference in a given context.

8. Correction of Inference Biases

Finally, we will briefly allude to an example of what we've called the "reflexive" stage in inference. Recall that this is a stage beyond Piaget's formal stage, reflecting the concerns of [25,28-31] that the Piagetan hierarchy ignores the ongoing development of cognition into adulthood.

Highly intelligent and self-aware adults may carry out reflexive cognition by explicitly reflecting upon their own inference processes and trying to improve them. An example is the intelligent improvement of uncertain-truth-value-manipulation formulas. It is well demonstrated that even educated humans typically make numerous errors in probabilistic reasoning [57,58]. Most people don't realize it and continue to systematically make these errors throughout their lives. However, a small percentage of individuals make an explicit effort to increase their accuracy in making probabilistic judgments by consciously endeavoring to internalize the rules of probabilistic inference into their automated cognition processes.

The same sort of issue exists even in an AI system such as Novamente which is explicitly based on probabilistic reasoning. PLN is founded on probability theory, but also contains a variety of heuristic assumptions that inevitably introduce a certain amount of error into its inferences. For example, the probabilistic deduction formula mentioned above embodies a heuristic independence assumption. Thus PLN contains an alternate deduction formula called the "concept geometry formula" [41] that is better in some contexts, based on the assumption that ConceptNodes embody concepts that are roughly spherically-shaped in attribute space. A highly advanced Novamente system could potentially augment the independence-based and concept-geometry-based deduction formulas with additional formulas of its own derivation, optimized to minimize error in various contexts. This is a simple and straightforward example of reflexive cognition--it illustrates the power accessible to a cognitive system that has formalized and reflected upon its own inference processes, and that possesses at least some capability to modify these.

9. Keeping Continuity in Mind

Continuity of mental stages, and the fact that a mind may appear to be in multiple stages of development simultaneously (depending upon the tasks being tested), are crucial to our theoretical formulations and we will touch upon them again here. Piaget attempted to address continuity with the creation of transitional "half stages". We prefer to observe that each stage feeds into the other and the end of one stage and the beginning of the next blend together.

The distinction between formal and post-formal, for example, seems to "merely" be the application of formal thought to oneself. However, the distinction between concrete and formal is "merely" the buildup to higher levels of complexity of the classification, task decomposition, and abstraction capabilities of the concrete stage. The stages represent general trends in ability on a continuous curve of development, not discrete states of mind which are jumped-into quantum style after enough "knowledge energy" builds-up to cause the transition.

Observationally, this appears to be the case in humans. People learn things gradually, and show a continuous development in ability, not a quick jump from ignorance to mastery. We believe that this gradual development of ability is the signature of genuine learning, and that prescriptively an AI system must be designed in order to have continuous and asymmetrical development across a variety of tasks in order to be considered a genuine learning system. While quantum leaps in ability may be possible in an AI system which can just "graft" new parts of brain onto itself (or an augmented human which may someday be able to do the same using implants), such acquisition of knowledge is not really learning. Grafting on knowledge does not build the cognitive pathways needed in order to actually learn. If this is the only mechanism available to an AI system to acquire new knowledge, then it is not really a learning system.

10. Our Theory: Applicability and Issues

Our theory is applicable to both humans, and AI Systems built upon uncertain inference systems which allow arbitrary complexity can both be described using our theory. Both humans and properly designed AI systems have all the characteristics of an uncertain inference system, and should exhibit all four stages of task capability. Humans have the first three already mastered, and may need AI systems to achieve the fourth. AI systems can more easily achieve the fourth once the first three are achieved, but need a lot of human help to get through the first three.

Though our theory is currently being further developed, and is not yet rigorously tested, we already have observed two issues with it which we will attempt to redress through our further theoretical and practical work.

So far, no AI system has made it to even the Concrete stage of development. However, our model gives guidelines for how to approach and chart this development. By defining the stages in ways which are equally applicable to AI systems and humans we hope to be able to give a framework for guiding the cognitive development of an AI as well as to help better describe human cognition for further analysis.

Also, humans may never proceed as far long into the reflexive postformal stage as AI systems. If we do, it may require the assistance of AI systems to help us understand and augment our biological hardware in ways we currently do not understand.

However, practices such as rational metasystemic thinking and irrational meditative practices may allow us to perform some amount of self-modification even without being able to directly alter our neural representations at a truly metasystemic level.

11. Conclusion

AI systems must *learn*, but they must also *develop*: and development in this sense takes place over a longer time scale than learning, and involves more fundamental changes in cognitive operation. Understanding the development of cognition is equally as important to AI as understanding the nature of cognition at any particular stage.

We have proposed a novel approach to defining developmental stages, in which internal properties of inference control systems are correlated with external learning capabilities, and have fleshed out the approach via giving a series of specific examples related to the Novamente AI Engine and the AGI-SIM world. Our future work with Novamente will involve teaching it to perform behaviors in the AGI-SIM world, progressing gradually through the developmental stages described here, using examples such as those given. Finally, we suspect that this approach to developmental psychology also has relevance beyond Novamente--most directly to other uncertain inference-based AI systems, and perhaps to developmental psychology in general.

References

[1] T. Shultz. *Computational Developmental Psychology*. Cambridge, MA: MIT Press, 2003.
[2] E. Thelen and L. Smith. *A Dynamic Systems Approach to the Development of Cognition and Action*. Cambridge, MA: MIT Press, 1994.
[3] J. A.S. Kelso. *Dynamic Patterns: The Self-Organization of Brain and Behavior*. Cambridge, MA: MIT Press, 1995.
[4] M. Howe and F. Rabinowitz. "Dynamic modeling, chaos, and cognitive development." *Journal of Experimental Child Psychology*, 1994, 58, 184-199.
[5] B. Goertzel, C. Pennachin, A. Senna, T. Maia, G. Lamacie. "Novamente: an integrative architecture for Artificial General Intelligence." *Proceedings of IJCAI 2003 Workshop on Cognitive Modeling of Agents*. Acapulco, Mexico, 2004.
[6] M. Looks, B. Goertzel and C. Pennachin. "Novamente: an integrative architecture for Artificial General Intelligence." *Proceedings of AAAI 2004 Symposium on Achieving Human-Level AI via Integrated Systems and Research,* Washington DC, 2004.
[7] B. Goertzel and Cassio Pennachin. "The Novamente AI engine." *Artificial General Intelligence*, Ed. by B. Goertzel and C. Pennachin, New York: Springer Verlag, 2005.
[8] B. Goertzel. *The Foundations of Emergent Digital Intelligence*. In preparation.
[9] J. Santore and S. Shapiro. "Crystal Cassie: Use of a 3-D gaming environment for a Cognitive Agent." In R. Sun, Ed., *Papers of the IJCAI 2003 Workshop on Cognitive Modeling of Agents and Multi-Agent Interactions*, IJCAII, Acapulco, Mexico, Aug. 9, 2003, 84-91.
[10] B. Inhelder and J. Piaget. *The Growth of Logical Thinking from Childhood to Adolescence*. New York: Basic Books, 1958.
[11] J. Piaget. *Structuralism*. New York: Harper & Row, 1970.
[12] J. Piaget. *Biology and Knowledge*. Chicago: University of Chicago Press, 1971.
[13] J. Piaget. "Piaget's theory." In P. Mussen (ed). *Handbook of Child Psychology*. 4th edition. Vol. 1. New York: Wiley, 1983
[14] J. Piaget. *Studies in Reflecting Abstraction*. Hove, UK: Psychology Press, 2001.
[15] S. Opper and H. Ginsburg. *Piaget's Theory of Development*. New York Prentice-Hall, 1987.
[16] L. Vygotsky. Mind in Society: The Development of Higher Psychological Processes. Cambridge, MA: Harvard University Press, 1978.
[17] Vygotsky. Thought and Language. Cambridge, MA: MIT Press, 1986.

[18] R., Gagne, L. Briggs, and W. Walter. *Principles of Instructional Design*, Fort Worth: Harcourt Brace Jovanovich College Publishers, 1992.
[19] J. Gibbs, "Kohlberg's moral stage theory: a Piagetian revision." *Human Development,* 1978, 22, 89-112.
[20] J. C. Gibbs, K. F. Widaman, and A. Colby. "Sociomoral reflection: Construction and validation of group-administrable measures." *Child Development*, 1982, 53, 895-910.
[21] J. Broughton. "Not beyond formal operations, but beyond Piaget." In M. Commons, F. Richards, and C. Armon (Eds.), *Beyond Formal Operations: Late Adolescent and Adult Cognitive Development*. New York: Praeger, 1984, 395-411.
[22] M.L. Commons and A. Pekker, *Hierarchical Complexity: A Formal Theory*, unpublished manuscript, submitted to J. Math. Psych., 2005.
[23] J. Pascual-Leone and J. Smith. "The encoding and decoding of symbols by children: a new experimental paradigm and neo-Piagetan model." *Journal of Experimental Child Psychology*, 1969, 8, 328-355.
[24] R. Gelman, E. Meck and S. Merkin. "Young children's numerical competence." *Cognitive Development*, 1986, 1, 1-29.
[25] K. Fischer. "A theory of cognitive development: control and construction of hierarchies of skills." *Psychological Review*, 1980, 87, 477-531.
[26] R. Case. *Intellectual development: Birth to adulthood.* New York: Academic Press, 1985.
[27] M.Bickhard. "Piaget on variation and selection models: Structuralism, logical necessity, and interactivism." *Human Development*, 1988, 31, 274-312.
[28] Arlin, P.K. "Cognitive development in adulthood: A fifth stage?" *Developmental Psychology,* 1975, 11, 602-606.
[29] M. Commons, F. Richards and D. Kuhn. "Systematic and metasystematic reasoning: a case for a level of reasoning beyond Piaget's formal operations." *Child Development*, 1982, 53, 1058-1069.
[30] K. Riegel. "Dialectic operations: the final phase of cognitive development." *Human Development*, 1973, 16, 346-370.
[31] H. Marchand. "Reflections on PostFormal Thought." In *The Genetic Epistemologist*, vol. 29, no. 3, 2001.
[32] K. Kitchener and P. King. "Reflective judgement: ten years of research." In M. Commons, et. Al. (Eds.), *Beyond Formal Operations Vol. 2: Models and Methods in the Study of Adolescent and Adult Thought*. New York: Praeger, 1990, 63-78.
[33] Commons, M., Trudeau, E.J., Stein, S.A., Richards, F.A., & Krause, S.R. "Hierarchical complexity of tasks shows the existence of developmental stages." *Developmental Review*, 1998, 18, 237-278.
[34] G. Engelbretsen and F. Sommers. "An invitation to formal reasoning." *The Logic of Terms*. Aldershot: Ashgate, 2000.
[35] L. Zadeh. Fuzzy sets as a basis for a theory of possibility. Fuzzy Sets and System, 1978, 1:3-28.
[36] I. Good. *The Estimation of Probabilities*. Cambridge, MA: MIT Press, 1986.
[37] R. Fung and C. Chong. Metaprobability and Dempster-Shafer in evidential reasoning. In L. Kanal and J. Lemmer (Eds.), *Uncertainty in Artificial Intelligence*, Amsterdam, Holland, 1986, 295-302.
[38] G. Shafer. A Mathematical Theory of Evidence. Princeton, NJ: Princeton University Press, 1976.
[39] H. Kyburg. Bayesian and non-Bayesian evidential updating. *Artificial Intelligence*, 1987, 31:271-293.
[40] P. Wang. *Non-Axiomatic Reasoning System.* PhD Thesis, Bloomington, IN: Indiana University, 1995.
[41] B. Goertzel, I. F. Goertzel, M. Iklé, A. Heljakka. *Probabilistic Logic. Networks* In preparation.
[42] D. MacKenzie. The automation of proof: A historical and sociological exploration. *IEEE Annals of the History of Computing*, 1995, 17(3), 7-29.
[43] B. Bollobas. *Modern Graph Theory*. New York: Springer 1998
[44] P. Walley. *Statistical Reasoning with Imprecise Probabilities*. New York: Chapman and Hall, 1991.
[45] S. Russell and P. Norvig P. *Artificial Intelligence: A Modern Approach*, Englewood Cliffs, NJ: Prentice Hall, 1995.
[46] B. Goertzel. Patterns, Hypergraphs and Embodied General Intelligence. Submitted to *Workshop on Human-Level Intelligence*, *WCCI 2006*, Vancouver.
[47] S. Baron-Cohen. *Mindblindness: An Essay on Autism and Theory of Mind.* Cambridge, MA: MIT Press, 1995.
[48] J. Fodor. *The Elm and the Expert*, Cambridge, MA: Bradford Books, 1994.
[49] J. Elman, E. Bates, M. Johnson, A. Karmiloff-Smith, D. Parisi, K. Plunkett. *Rethinking Innateness: A Connectionist Perspective on Development*. MIT Press, 1997.
[50] D. Davidson. *Inquiries into Truth and Interpretation*, Oxford: Oxford University Press, 1984.
[51] D. Dennett. *The Intentional Stance*, Cambridge, MA: MIT Press, 1987.
[52] D. Premack and G. Woodruff. "Does the chimpanzee have a theory of mind?" *Behavioral and Brain Sciences*, 1978 1, 515-526.
[53] M. Tomasello and J. Call. *Primate Cognition*. New York: Oxford University Press, 1997.
[54] R. W. Byrne and A. Whiten. *Machiavellian Intelligence*. Clarendon Press, 1988.

[55] S. Carey. *Conceptual Change in Childhood*, Cambridge, MA: MIT Press, 1985.
[56] H. Wellman. *The Child's Theory of Mind*, Cambridge, MA: MIT Press, 1990.
[57] R. Gordon. Folk Psychology as Simulation. *Mind and Language,* 1986, 1, 158-171.
[58] P. Wason. Reasoning. In B. M. Foss (Ed.), *New Horizons in Psychology*. Middlesex, UK: Penguin, 1966
[59] T. Gilovich, D. Griffin & D. Kahneman, (Eds.). *Heuristics and biases: The psychology of intuitive judgment*. Cambridge, UK: Cambridge University Press, 2002

Advances in Artificial General Intelligence: Concepts, Architectures and Algorithms
B. Goertzel and P. Wang (Eds.)
IOS Press, 2007

© 2007 The authors and IOS Press. All rights reserved.

Indefinite Probabilities for General Intelligence[1]

Matthew IKLÉ[a], Ben GOERTZEL[b] and Izabela GOERTZEL[b]
[a]*Adams State College, Alamosa, Colorado and Novamente LLC*
[b]*Novamente LLC*

Abstract. The creation of robust mechanisms for uncertain inference is central to the development of Artificial General Intelligence systems. While probability theory provides a principled foundation for uncertain inference, the mathematics of probability theory has not yet been developed to the point where it is possible to handle every aspect of the uncertain inference process in practical situations using rigorous probabilistic calculations. Due to the need to operate within realistic computational resources, probability theory presently requires augmentation with heuristics in order to be pragmatic for general intelligence (as well as for other purposes such as large-scale data analysis).

The authors have been involved with the creation of a novel, general framework for pragmatic probabilistic inference in an AGI context, called Probabilistic Logic Networks (PLN). PLN integrates probability theory with a variety of heuristic inference mechanisms; it encompasses a rich set of first-order and higher-order inference rules, and it is highly flexible and adaptive, and easily configurable. This paper describes a single, critical aspect of the PLN framework, which has to with the quantification of uncertainty. In short, it addresses the question: *What should an uncertain truth value be, so that a general intelligence may use it for pragmatic reasoning?*

We propose a new approach to quantifying uncertainty via a hybridization of Walley's theory of imprecise probabilities and Bayesian credible intervals. This "indefinite probability" approach provides a general method for calculating the "weight-of-evidence" underlying the conclusions of uncertain inferences. Moreover, both Walley's imprecise beta-binomial model and standard Bayesian inference can be viewed mathematically as special cases of the more general indefinite probability model. Via exemplifying the use of indefinite probabilities in a variety of PLN inference rules (including exact and heuristic ones), we argue that this mode of quantifying uncertainty may be adequate to serve as an ingredient of powerful artificial general intelligence.

Introduction

As part of our ongoing work on the Novamente artificial general intelligence (AGI) system, we have developed a logical inference system called Probabilistic Logic Networks (PLN), designed to handle the various forms of uncertain inference that may confront a general intelligence – including reasoning based on uncertain knowledge, and/or reasoning leading to uncertain conclusions (whether from certain or uncertain

[1] The authors would like to thank Pei Wang for his very detailed and insightful comments on an earlier draft, which resulted in significant improvements to the paper.

knowledge). Among the general high-level requirements underlying the development of PLN have been the following:

- To enable uncertainty-savvy versions of all known varieties of logical reasoning, including for instance higher-order reasoning involving quantifiers, higher-order functions, and so forth.
- To reduce to crisp "theorem prover" style behavior in the limiting case where uncertainty tends to zero.
- To encompass inductive and abductive as well as deductive reasoning.
- To agree with probability theory in those reasoning cases where probability theory, in its current state of development, provides solutions within reasonable calculational effort based on assumptions that are plausible in the context of real-world data.
- To gracefully incorporate heuristics not explicitly based on probability theory, in cases where probability theory, at its current state of development, does not provide adequate pragmatic solutions.
- To provide "scalable" reasoning, in the sense of being able to carry out inferences involving at least billions of premises. Of course, when the number of premises is fewer, more intensive and accurate reasoning may be carried out.
- To easily accept input from, and send input to, natural language processing software systems.

PLN implements a wide array of first-order and higher-order inference rules including (but not limited to) deduction, Bayes' Rule, unification, intensional and extensional inference, belief revision, induction, and abduction. Each rule comes with uncertain truth-value formulas, calculating the truth-value of the conclusion from the truth-values of the premises. Inference is controlled by highly flexible forward and backward chaining processes able to take feedback from external processes and thus behave adaptively.

The development of PLN has taken place under the assumption that probability theory is the "right" way to model uncertainty (more on this later). However, the mathematics of probability theory (and its interconnection with other aspects of mathematics) has not yet been developed to the point where it is feasible to use fully rigorous, explicitly probabilistic methods to handle every aspect of the uncertain inference process.

One of the major issues with probability theory as standardly utilized involves the very quantification of the uncertainty associated with statements that serve as premises or conclusions of inference. Using a single number to quantify the uncertainty of a statement is often not sufficient, a point made very eloquently by Wang ([1]), who argues in detail that the standard Bayesian approach does not offer any generally viable way to assess or reason about the "second-order uncertainty" involved in a given probability assignment. Probability theory provides richer mechanisms than this: one may assign a probability distribution to a statement, instead of a single probability value. But what if one doesn't have the data to fill in a probability distribution in detail? What is the (probabilistically) best approach to take in the case where a single number is not enough but the available data doesn't provide detailed distributional information? Current probability theory does not address this issue adequately. Yet this is a critical question if one wants to apply probability theory in a general intelligence context. In short, one needs methods of quantifying uncertainty at an intermediate level of detail between single probability numbers and fully known probability distributions. This is what we mean by the question: *What should an uncertain truth-value be, so that a general intelligence may use it for pragmatic reasoning?*

1. From Imprecise Probabilities to Indefinite Probabilities

Walley's ([2]) theory of imprecise probabilities seeks to address this issue, via defining interval probabilities, with interpretations in terms of families of probability distributions. The idea of interval probabilities was originally introduced by Keynes ([3]), but Walley's version is more rigorous, grounded in the theory of envelopes of probability distributions. Walley's intervals, so-called "imprecise probabilities," are satisfyingly natural and consistent in the way they handle uncertain and incomplete information. However, in spite of a fair amount of attention over the years, this line of research has not yet been developed to the point of yielding robustly applicable mathematics.

Using a parametrized envelope of (beta-distribution) priors rather than assuming a single prior as would be typical in the Bayesian approach, Walley ([2,4]) concludes that it is plausible to represent probabilities as intervals of the form $\left[\frac{m}{n+k}, \frac{m+k}{n+k}\right]$. In this formula, n represents the total number of observations, m represents the number of positive observations, and k is a parameter that Walley calls s and derives as a parameter of the beta distribution. Walley calls this parameter the learning parameter, while we will refer to it as the skepticism parameter. Note that the width of the interval of probabilities is inversely related to the number of observations n, so that the more evidence one has, the narrower the interval. The parameter k determines how rapidly this narrowing occurs. An interval of this sort is what Walley calls an "imprecise probability."

Walley's approach comes along with a host of elegant mathematics including a Generalized Bayes' Theorem. However it is not the only approach to interval probabilities. For instance, one alternative is Weichselberger's ([5]) axiomatic approach, which works with sets of probabilities of the form $[L,U]$ and implies that Walley's generalization of Bayes' rule is not the correct one.

One practical issue with using interval probabilities like Walley's or Weichselberger's in the context of probabilistic inference rules (such as those used in PLN) is the pessimism implicit in interval arithmetic. If one takes traditional probabilistic calculations and simplistically replaces the probabilities with intervals, then one finds that the intervals rapidly expand to [0,1]. This fact simply reflects the fact that the intervals represent "worst case" bounds. This same problem also affects Walley's and Weichselberger's more sophisticated approaches, and other approaches in the imprecise probabilities literature. The indefinite probabilities approach presented here circumvents these practical problems via utilizing interval probabilities that have a different sort of semantics – closely related to, but not the same as, those of Walley's interval probabilities.

Indefinite probabilities, as we consider them here, are represented by quadruples of the form ([L,U],b,k) – thus, they contain two additional numbers beyond the [L,U] interval truth values proposed by Keynes, and one number beyond the ([L,U],k) formalism proposed by Walley. The semantics involved in assigning such a truth value to a statement S is, roughly, "I assign a probability of b to the hypothesis that, after I have observed k more pieces of evidence, the truth value I assign to S will lie in the interval [L,U]." In the practical examples presented here we will hold k constant and thus will deal with truth value triples ([L,U],b).

The inclusion of the value b, which defines the credibility level according to which [L,U] is a credible interval (for hypothesized future assignments of the probability of S, after observing k more pieces of evidence), is what allows our intervals to generally remain narrower than those produced by existing imprecise probability approaches. If b = 1, then our approach essentially reduces to imprecise probabilities, and in pragmatic inference contexts tends to produce intervals [L,U] that approach [0,1]. The use of b < 1 allows the inferential production of narrower intervals, which are more useful in a real-world inference context.

In practice, to execute inferences using indefinite probabilities, we make heuristic distributional assumptions, assuming a "second order" distribution which has [L,U] as a (100*b)% credible interval, and then "first order" distributions whose means are drawn from the second-order distribution. These distributions are to be viewed as heuristic approximations intended to estimate unknown probability values existing in hypothetical future situations. The utility of the indefinite probability approach may be dependent on the appropriateness of the particular distributional assumptions to the given application situation. But in practice we have found that a handful of distributional forms seem to suffice to cover common-sense inferences (beta and bimodal forms seem good enough for nearly all cases; and here we will only give examples covering the beta distribution case).

Because the semantics of indefinite probabilities is different from that of ordinary probabilities, or imprecise probabilities, or for example NARS truth values, it is not possible to say objectively that any one of these approaches is "better" than the other one, as a mathematical formalism. Each approach is better than the others at mathematically embodying its own conceptual assumptions. From an AGI perspective, the value of an approach to quantifying uncertainty lies in its usefulness when integrated with a pragmatic probabilistic reasoning engine. The bulk of this paper will be concerned with showing how indefinite probabilities behave when incorporated in the logical reasoning rules utilized in the PLN inference framework, a component of the Novamente AI Engine. While complicated and dependent on many factors, this is nevertheless the sort of evaluation that we consider most meaningful.

Section 2 deals with the conceptual foundations of indefinite probabilities, clarifying their semantics in the context of Bayesian and frequentist philosophies of probability. Section 3 outlines the pragmatic computational method we use for doing probabilistic and heuristic inference using indefinite probabilities. Section 4 presents a number of specific examples involving using indefinite probabilities within single inference steps within the PLN inference framework.

2. The Semantics of Uncertainty

The main goal of this paper is to present indefinite probabilities as a pragmatic tool for uncertain inference, oriented toward utilization in AGI systems. Before getting practical, however, we will pause in this section to discuss the conceptual, semantic foundations of the "indefinite probability" notion. In the course of developing the indefinite probabilities approach, we found that the thorniest aspects lay not in the mathematics or software implementation, but rather in the conceptual interpretation of the truth values and their roles in inference.

In the philosophy of probability, there are two main approaches to interpreting the meaning of probability values, commonly labeled frequentist and Bayesian ([6]). There are many shades of meaning to each interpretation, but the essential difference is easy to understand. The frequentist approach holds that a probability should be interpreted as the limit of the relative frequency of an event-category, calculated over a series of events as the length of the series tends to infinity. The subjectivist or Bayesian approach holds that a probability should be interpreted as the degree of belief in a statement, held by some observer; or in other words, as an estimate of how strongly an observer believes the evidence available to him supports the statement in question. Early proponents of the subjectivist view were Ramsey ([7]) and de Finetti ([8]), who argued that for an individual to display self-consistent betting behavior they would need to assess degrees of belief according to the laws of probability theory. More recently Cox's Theorem ([9]) and related mathematics ([10]) have come into prominence as providing a rigorous foundation for subjectivist probability. Roughly speaking, this mathematical work shows that if the observer assessing subjective probabilities is to be logically consistent, then their plausibility estimates must obey the standard rules of probability.

From a philosophy-of-AI point of view, neither the frequentist nor the subjectivist interpretations, as commonly presented, is fully satisfactory. However, for reasons to be briefly explained here, we find the subjectivist interpretation more acceptable, and will consider indefinite probabilities within a subjectivist context, utilizing relative frequency calculations for pragmatic purposes but giving them an explicitly subjectivist rather than frequentist interpretation.

The frequentist interpretation is conceptually problematic in that it assigns probabilities only in terms of limits of sequences, not in terms of finite amounts of data. Furthermore, it has well-known difficulties with the assignment of probabilities to unique events that are not readily thought of as elements of ensembles. For instance, what was the probability, in 1999, of the statement S holding that "A great depression will be brought about by the Y2K problem"? Yes, this probability can be cast in terms of relative frequencies in various ways. For instance, one can define it as a relative frequency across a set of hypothetical "possible worlds": across all possible worlds similar to our own, in how many of them did the Y2K problem bring about a great depression? But it's not particularly natural to assume that this is what an intelligence must do in order to assign a probability to S. It would be absurd to claim that, in order to assign a probability to S, an intelligence must explicitly reason in terms of an ensemble of possible worlds. Rather, the claim must be that whatever reasoning a mind does to evaluate the probability of S may be implicitly interpreted in terms of possible worlds. This is not completely senseless, but is a bit of an irritating conceptual stretch.

The subjectivist approach, on the other hand, is normally conceptually founded either on rational betting behaviors or on Cox's Theorem and its generalizations, both of which are somewhat idealistic.

No intelligent agent operating within a plausible amount of resources can embody fully self-consistent betting behavior in complex situations. The irrationality of human betting behavior is well known; to an extent this is due to emotional reasons, but there are also practical limitations on the complexity of the situation in which any finite mind can figure out the correct betting strategy.

And similarly, it is too much to expect any severely resource-constrained intelligence to be fully self-consistent in the sense that the assumptions of Cox's theorem require. In order to use Cox's Theorem to justify the use of probability theory by practical intelligences, it seems to us, one would need to take another step beyond Cox, and argue that if an AI system is going to have a "mostly sensible" measure of plausibility (i.e. if its deviation from Cox's axioms are not too great), then its intrinsic plausibility measure must be *similar to* probability. We consider this to be a viable line of argument, but will pursue this point in another paper – to enlarge on such matters here would take us too far afield.

Walley's approach to representing uncertainty is based explicitly on a Bayesian, subjectivist interpretation; though whether his mathematics has an alternate frequentist interpretation is something he has not explored, to our knowledge. Similarly, our approach here is to take a subjectivist perspective on the foundational semantics of indefinite probabilities (although we don't consider this critical to our approach; quite likely it could be given a frequentist interpretation as well.) Within our basic subjectivist interpretation, however, we will frequently utilize relative frequency calculations when convenient for pragmatic reasoning. This is conceptually consistent because within the subjectivist perspective, there is still a role for relative frequency calculations, so long as they are properly interpreted.

Specifically, when handling a conditional probability $P(A|B)$, it may be the case that there is a decomposition $B = B_1 +... + B_n$ so that the B_i are mutually exclusive and equiprobable, and each of $P(A|B_i)$ is either 0 or 1. In this case the laws of probability tell us $P(A|B) = P(A|B_1) P(B_1| B) + ... + P(A|B_n) P(B_n|B) = (P(A|B_1) + ... + P(A|B_n))/n$, which is exactly a relative frequency. So, in the case of statements that are decomposable in this sense, the Bayesian interpretation implies a relative frequency based interpretation (but not a "frequentist" interpretation in the classical sense). For decomposable statements, plausibility values may be regarded as the means of probability distributions, where the distributions may be derived via subsampling (sampling subsets C of $\{B_1,...,B_n\}$, calculating $P(A|C)$ for each subset, and taking the distribution of these values; as in the statistical technique known as bootstrapping). In the case of the "Y2K" statement and other similar statements regarding unique instances, one option is to think about decomposability across possible worlds, which is conceptually controversial.

2.1. Indefinite Probability

We concur with the subjectivist maxim that a probability can usefully be interpreted as an estimate of the plausibility of a statement, made by some observer. However, we suggest introducing into this notion a more careful consideration of the role of evidence in the assessment of plausibility. We introduce a distinction that we feel is critical, between

- the ordinary (or "definite") plausibility of a statement, interpreted as the degree to which the evidence already (directly or indirectly) collected by a particular observer supports the statement.
- the "indefinite plausibility" of a statement, interpreted as the degree to which the observer believes that the overall body of evidence potentially available to him supports the statement.

The indefinite plausibility is related to the ordinary plausibility, but also takes into account the potentially limited nature of the store of evidence collected by the observer at a given point in time. While the ordinary plausibility is effectively represented as a single number, the indefinite plausibility is more usefully represented in a more complex form. We suggest to represent an indefinite plausibility as a quadruple $([L,U],b,k)$, which when attached to a statement S has the semantics "I assign an ordinary plausibility of b to the statement that 'Once k more items of evidence are collected, the ordinary plausibility of the statement S will lie in the interval [L,U]'". Note that indefinite plausibility is thus defined as "second order plausibility" – a plausibility of a plausibility.

As we shall see in later sections of the paper, for most computational purposes it seems acceptable to leave the parameter k in the background, assuming it is the same for both the premises and the conclusion of an inference. So in the following we will mainly speak of indefinite probabilities as $([L,U],b)$ triples, for sake of simplicity. The possibility does exist, however, that in future work, inference algorithms will be designed that utilize k explicitly.

Now, suppose we buy the Bayesian argument that ordinary plausibility is best represented in terms of probability. Then it follows that indefinite plausibility is best represented in terms of second-order probability, i.e. as "I assign probability b to the statement that 'Once k more items of evidence have been collected, the probability of the truth of S based on this evidence will lie in the interval $[L,U]$'".

2.1.1. An Interpretation in Term of Betting Behavior

To justify the above definition of indefinite probability more formally, one approach is to revert to betting arguments of the type made by de Finetti in his work on the foundations of probability. As will be expounded below, for computational purposes, we have taken a pragmatic frequentist approach, based on underlying distributional assumptions. However, for purposes of conceptual clarity, a more subjectivist de Finetti style justification is nevertheless of interest. So, in this subsection, we will describe a "betting scenario" that leads naturally to a definition of indefinite probabilities.

Suppose we have a category C of discrete events, e.g. a set of tosses of a certain coin which has heads on one side and tails on the other.

Next, suppose we have a predicate S, which is either True or False (boolean values) for each event within the above event-category. C. For example, if C is a set of tosses of a certain coin, then S could be the event "Heads". S is a function from events into Boolean values.

If we have an agent A, and the agent A has observed the evaluation of S on n different events, then we will say that n is the amount of evidence that A has observed regarding S; or we will say that A has made n observations regarding S.

Now consider a situation with three agents: the House, the Gambler, and the Meta-gambler.

As the name indicates, the House is going to run a gambling operation, involving generating repeated events in category C, and proposing bets regarding the outcome of future events in C.

More interestingly, House is also going to propose bets to the Meta-gambler, regarding the behavior of the Gambler.

Specifically, suppose the House behaves as follows.

After the Gambler makes n observations regarding S, House offers Gambler the opportunity to make what we'll call a "de Finetti" type bet regarding the outcome of the next observation of S.That is, House offers Gambler the opportunity:

> "You must set the price of a promise to pay $1 if the next observation of S comes out True, and $0 if there it does not. You must commit that I will be able to choose either to buy such a promise from you at the price you have set, or require you to buy such a promise from me. In other words: you set the odds, but I decide which side of the bet will be yours."

Assuming the Gambler does not want to lose money, the price Gambler sets in such a bet, is the "operational subjective probability" that Gambler assigns that the next observation of S will come out True.

As an aside, House might also offer Gambler the opportunity to bet on sequences of observations, e.g. it might offer similar "de Finetti" price-setting opportunities regarding predicates like "The next 5 observations of S made will be in the ordered pattern (True, True, True, False, True)." In this case, things become interesting if we suppose Gambler thinks that: For each sequence Z of {True, False} values emerging from repeated observation of S, any permutation of Z has the same (operational subjective) probability as Z. Then, Gambler thinks that the series of observations of S is "exchangeable", which means intuitively that S's subjective probability estimates are really estimates of the "underlying probability of S being true on a random occasion." Various mathematical conclusions follow from the assumption that Gambler does not want to lose money, combined with the assumption that Gambler believes in exchangeability.

Next, let's bring Meta-gambler into the picture. Suppose that House, Gambler and Meta-gambler have all together been watching n observations of S. Now, House is going to offer Meta-gambler a special opportunity. Namely, he is going to bring Meta-gambler into the back room for a period of time. During this period of time, House and Gambler will be partaking in a gambling process involving the predicate S.

Specifically, while Meta-gambler is in the back room, House is going to show Gambler k new observations of S. Then, after the k'th observation, House is going to come drag Meta-gambler out of the back room, away from the pleasures of the flesh and back to the place where gambling on S occurs.

House then offers Gambler the opportunity to set the price of yet another de-Finetti style bet on yet another observation of S.

Before Gambler gets to set his price, though, Meta-gambler is going to be given the opportunity of placing a bet regarding what price Gambler is going to set.

Specifically, House is going to allow Meta-gambler to set the price of a de Finetti style bet on a proposition of Meta-gambler's choice, of the form:

Q = "Gambler is going to bet an amount p that lies in the interval [L,U]"

For instance Meta-gambler might propose

"Let Q be the proposition that Gambler is going to bet an amount lying in [.4, .6] on this next observation of S. I'll set at 30 cents the price of a promise defined as follows: To pay $1 if Q comes out True, and $0 if it does not. I will commit that you will be able to choose either to buy such a promise from me at this price, or require me to buy such a promise from you."

I.e., Meta-Gambler sets the price corresponding to Q, but House gets to determine which side of the bet to take.

Let us denote the price set by Meta-gambler as b; and let us assume that Meta-gambler does not want to lose money.

Then, b is Meta-gambler's subjective probability assigned to the statement that:

"Gambler's subjective probability for the next observation of S being True lies in [L,U]."

But, recall from earlier that the indefinite probability

<L,U,b,k> attached to S means that:

"The estimated odds are b that after k more observations of S, the estimated probability of S will lie in [L,U]."

or in other words

"[L,U] is a b-level credible interval for the estimated probability of S after k more observations."

In the context of an AI system reasoning using indefinite probabilities, there is no explicit separation between the Gambler and the Meta-gambler; the same AI system makes both levels of estimate. But this is of course not problematic, so long as the two components (first-order probability estimation and b-estimation) are carried out separately.

One might argue that this formalization in terms of betting behavior doesn't really add anything practical to the indefinite probabilities framework as already formulated. At minimum, however, it does make the relationship between indefinite probabilities and the classical subjective interpretation of probabilities quite clear.

2.1.2. A Pragmatic Frequentist Interpretation

Next, it is not hard to see how the above-presented interpretation of an indefinite plausibility can be provided with an alternate justification in relative frequency terms, in the case where one has a statement S that is decomposable in the sense described above. Suppose that, based on a certain finite amount of evidence about the frequency of a statement S, one wants to guess what one's frequency estimate will be once one has seen a lot more evidence. This guessing process will result in a probability distribution across frequency estimates – which may itself be interpreted as a frequency via a "possible worlds" interpretation. One may think about "the frequency, averaged across all possible worlds, that we live in a world in which the observed frequency of S after k more observations will lie in interval I." So, then, one may interpret ([L,U], b, N) as meaning "b is the frequency of possible worlds in which the observed frequency of S, after I've gathered k more pieces of evidence, will lie in the interval [L,U]."

This interpretation is not as conceptually compelling as the betting-based interpretation given above – because bets are real things, whereas these fictitious possible worlds are a bit slipperier. However, we make use of this frequency-based interpretation of indefinite probabilities in the practical computational implementation of indefinite probability presented in the following sections – without, of course, sacrificing the general Bayesian interpretation of the indefinite probability approach. In the end, we consider the various interpretations of probability to be in the main complementary rather than contradictory, providing different perspectives on the same very useful mathematics.

Moving on, then: To adopt a pragmatic frequency-based interpretation of the second-order plausibility in the definition of indefinite plausibility, we interpret "I assign probability b to the statement that 'Once k more items of evidence are collected, the probability of the truth of S based on this evidence will lie in the interval $[L,U]$'" to mean "b is the frequency, across all possible worlds in which I have gathered k more items of evidence about S, of worlds in which the statement 'the estimated probability of S lies in the interval $[L,U]$' is true". This frequency-based interpretation allows us to talk about a probability distribution consisting of probabilities assigned to values of 'the estimated probability of S', evaluated across various possible worlds. This probability distribution is what, in the later sections of the paper, we call the "second-order distribution." For calculational purposes, we assume a particular distributional form for this second-order distribution.

Next, for the purpose of computational implementation, we make the heuristic assumption that the statement S under consideration is decomposable, so that in each possible world, "the estimated probability of S" may be interpreted as the mean of a probability distribution. For calculational purposes, in our current implementation we assume a particular distributional form for these probability distributions, which we refer to as "the first-order distributions."

The adoption of a frequency-based interpretation for the second-order plausibility seems hard to avoid if one wants to do practical calculations using the indefinite probabilities approach. On the other hand, the adoption of a frequency-based interpretation for the first-order plausibilities is an avoidable convenience, which is appropriate only in some situations. We will discuss below how the process of reasoning using indefinite probabilities can be simplified, at the cost of decreased robustness, in cases where decomposability of the first order probabilities is not a plausible assumption.

So, to summarize, in order to make the indefinite probabilities approach computationally tractable, we begin by restricting attention to some particular family D of probability distributions. Then, we interpret an interval probability attached to a statement as an assertion that: "There is probability b that the subjective probability of the statement, after I have made k more observations, will appear to be drawn from a distribution with a mean in this interval."

Then, finally, given this semantics and a logical inference rule, one can ask questions such as: "If each of the premises of my inference corresponds to some interval, so that there is probability b that after k more observations the distribution governing the premise will appear to have a mean in that interval; then, what is an interval so that b of the family of distributions of the conclusion have means lying in that interval?"[2] We may then give this final interval the interpretation that, after k more observations, there

[2] In fact, this is a minor oversimplification, because the credibility value b may be different for each premise and for the conclusion.

is a probability b that the conclusion of the inference will appear to lie in this final interval. (Note that, as mentioned above, the parameter k essentially "cancels out" during inference, so that one doesn't need to explicitly account for it during most inference operations, so long as one is willing to assume it is the same in the premises and the conclusion.)

In essence, this strategy merges the idea of imprecise probabilities with the Bayesian concept of credible intervals; thus the name "indefinite probabilities" ("definite" having the meaning of "precise," but also the meaning of "contained within specific boundaries" – Walley's probabilities are contained within specific boundaries, whereas ours are not).

3. Varieties of Uncertain Truth Value

We have discussed above the semantic foundations of truth values of the form $([L,U],b,k)$. Before proceeding further to discuss inference with these objects, it is worth pausing to note that, within the PLN inference framework, this is one of several forms of truth value object utilized. In general, the forms of truth value objects currently utilized in PLN are:

- IndefiniteTruthValues (as already discussed).
- SimpleTruthValues (to be described below).
- DistributionalTruthValues (which, in their most general form, involve maintaining a whole probability distribution of possible probability distributions attached to a statement).

SimpleTruthValues take two, equivalent forms:

- (s,w), where s is the mean strength, the best single-number probability assignable to the statement; w is the "weight of evidence", a number in [0,1] that tells you, qualitatively, how much you should believe the strength estimate.
- (s,n), where s is the mean strength; n is the "count", a number > 0 telling you, qualitatively, the total amount of evidence that was evaluated in order to assess s.

Borrowing from Pei Wang's NARS system ([11,12]), the weight-of-evidence and count in SimpleTruthValues are related via $w = \frac{n}{n + k1}$, where $k1$ is the system-wide skepticism parameter. Weight-of evidence is essentially a normalized version of the count. Similarly, n and [L,U] may be interrelated by a heuristic formula, n = k1 (1 – W)/W where W = U – L is the width of the credible interval. More rigorous formulas interrelating n and ([L,U],b,k) may be formulated if one makes specific distributional assumptions (e.g. assuming an underlying Bernoulli process), but these lead to qualitatively similar results to the above heuristic formula, and will be discussed in a later publication.

Note also that if [L,U] is of the form [m/(n + k1), (m + k1)/(n + 1k)], then k1 is interpretable as the number of hypothetical additional pieces of evidence being considered to obtain the mean-estimates in [L,U]. That is, in this case the parameter k1 associated with the ([L,U],b) truth value may be interpreted as equal to the parameter k used in the definition of indefinite truth values.

As hinted above, however, the above descriptions mask the complexity of the actual truth-value objects (except in the SimpleTruthValue case, which actually is simple). In the indefinite probabilities approach, in practice, each IndefiniteTruthValue object is also endowed with three additional parameters:

- an indicator of whether $[L,U]$ should be considered as a symmetric or asymmetric credible interval.
- a family of "second-order" distributions, used to govern the second-order plausibilities described above.
- a family of "first-order" distributions, used to govern the first-order plausibilities described above.

Combined with these additional parameters, each truth-value object essentially provides a compact representation of a single second-order probability distribution with a particular, complex structure.

4. Inference Using Indefinite Probabilities

We now describe the general process according to which a PLN inference formula is evaluated using indefinite probabilities. This process contains three basic steps. We will first present these steps in an abstract way; and then, in the following section, exemplify them in the context of specific probabilistic inference formulas corresponding to inference rules like term logic and modus ponens deduction, Bayes' Rule, and so forth.

In general, the process is the same for any inference formula that can be thought of as taking in premises labeled with probabilities, and outputting a conclusion labeled with a probability. The examples given here will pertain to single inference steps, but the same process may be applied holistically to a series of inference steps, or most generally an "inference tree" of inference steps, each building on conclusions of prior steps.

Step One in the indefinite probabilistic inference process is as follows. Given intervals, $[L_i,U_i]$, of mean premise probabilities, we first find a distribution from the "second-order distribution family" supported on $[L1_i,U1_i]\supset[L_i,U_i]$, so that these means have $[L_i,U_i]$ as $(100\cdot b_i)$% credible intervals. The intervals $[L1_i,U1_i]$ are either of the form $\left[\frac{m}{n+k},\frac{m+k}{n+k}\right]$, when the "interval-type" parameter associated with the premise is asymmetric; or are such that both of the intervals $[L1_i,L_i]$ and $[U_i,U1_i]$ each have probability mass $b_i/2$, when interval-type is symmetric.

Next, in Step Two, we use Monte-Carlo methods on the set of premise truth-values to generate final distribution(s) of probabilities as follows. For each premise, we randomly select n_1 values from the ("second-order") distribution found in step 1. These n_1 values provide the values for the means for our first-order distributions. For each of our n_1 first-order distributions, we select n_2 values to represent the first-order distribution. We apply the applicable inference rule (or tree of inference rules) to each of the n_2 values for each first-order distribution to generate the final distribution of first-order distributions of probabilities. We calculate the mean of each distribution and then – in

Step Three of the overall process – find a $(100 \cdot b_i)$% credible interval, $[L_f, U_f]$, for this distribution of means.

When desired, we can easily translate our final interval of probabilities, $[L_f, U_f]$, into a final, (s_f, n_f, b_f) triple of strength, count, and credibility levels, as outlined above.

Getting back to the "Bayesianism versus frequentism" issues raised in the previous section, if one wished to avoid the heuristic assumption of decomposability regarding the first-order plausibilities involved in the premises, one would replace the first-order distributions assumed in Step Two with Dirac delta functions, meaning that no variation around the mean of each premise plausibility would be incorporated. This would also yield a generally acceptable approach, but would result in overall narrower conclusion probability intervals, and we believe would in most cases represent a step away from realism and robustness.

4.1. The Procedure in More Detail

We now run through the above three steps in more mathematical detail.

In Step One of the above procedure, we assume that the mean strength values follow some given initial probability distribution $g_i(s)$ with support on the interval $[L1_i, U1_i]$. If the interval-type is specified as asymmetric, we perform a search until we find values *k2* so that $\int_{L_i}^{U_i} g_i(s)ds = b$, where $L_i = \frac{m_i}{n_i + k2}$ and $U_i = \frac{m_i + k2}{n_i + k2}$. If the interval-type is symmetric, we first ensure, via parameters, that each first-order distribution is symmetric about its mean s_i, set $L_i = s_i - d$, $U_i = s_i + d$ and perform a search for d to ensure that $\int_{L_i}^{U_i} g_i(s)ds = b$. In either case, each of the intervals $[L_i, U_i]$ will be a $(100 \cdot b)$% credible interval for the distribution $g_i(s)$.

We note that we may be able to obtain the appropriate credible intervals for the distributions $g_i(s)$ only for certain values of b. For this reason, we say that a value b is truth-value-consistent whenever it is feasible to find $(100 \cdot b)$% credible intervals of the appropriate type.

In Step Two of the above procedure, we create a family of distributions, drawn from a pre-specified set of distributional forms, and with means in the intervals $[L_i, U_i]$. We next apply Monte Carlo search to form a set of randomly chosen "premise probability tuples". Each tuple is formed via selecting, for each premise of the inference rule, a series of points drawn at random from randomly chosen distributions in the family. For each randomly chosen premise probability tuple, the inference rule is executed. And then, in Step Three, to get a probability value s_f for the conclusion, we take the mean of this distribution. Also, we take a credible interval from this final distribution, using a pre-specified credibility level b_f, to obtain an interval for the conclusion $[L_f, U_f]$.

When the interval-type parameter is set to asymmetric, to find the final count value n_f, note that *m=ns*. Then, since the skepticism parameter *k1* is a system parameter, we need only solve the following equation for the count n_f:

$$\int_{\frac{n_f s_f}{n_f + k1}}^{\frac{n_f s_f + k1}{n_f + k1}} g_f(s)ds = b$$

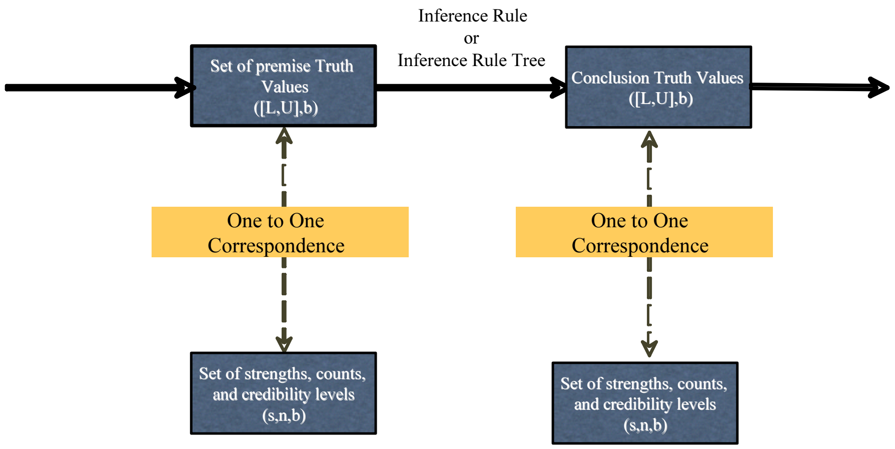

Inference Rule
or
Inference Rule Tree
Set of premise Truth Values ([L,U],b)
Conclusion Truth Values ([L,U],b)
One to One Correspondence
One to One Correspondence
Set of strengths, counts, and credibility levels (s,n,b)
Set of strengths, counts, and credibility levels (s,n,b)

Diagram 1.

For symmetric interval-types, we use the heuristic $n = c(1 - W)/W$ where W, the interval width, is $W = U_f - L_f$.

5. A Few Detailed Examples

In this section we report some example results obtained from applying the indefinite probabilities approach in the context of simple inference rules, using both symmetric and asymmetric interval-types.

Comparisons of results on various inference rules indicated considerably superior results in all cases when using the symmetric intervals. As a result, we will report results for five inference rules using the symmetric rules; while we will report the results using the asymmetric approach for only one example (term logic deduction).

5.1. A Detailed Bet-Binominal Example for Bayes' Rule

First we will treat Bayes' Rule, a paradigm example of an uncertain inference rule – which however is somewhat unrepresentative of inference rules utilized in PLN, due to its non-heuristic, exactly probabilistic nature.

The beta-binomial model is commonly used in Bayesian inference, partially due to the conjugacy of the beta and binomial distributions. In the context of Bayes' rule,

Table 1. Intervals for Crediblity Level 0.90

EVENT	**[L, U]**	**[L1, U1]**
G	$\left[\frac{10}{21},\frac{12}{21}\right]$ = [.476190, 0.571429]	[0.419724, 0.627895]
R	$\left[\frac{8}{21},\frac{12}{21}\right]$ = [0.380952, 0.571429]	[0.26802, 0.684361]
B|G	[0.3, 0.7]	[0.0628418, 0.937158]
B|R	[0.06, 0.14]	[0.0125683, 0.187432]

Table 2. Intervals for Crediblity Level 0.95

EVENT	**[L, U]**	**[L1, U1]**
G	$\left[\frac{10}{21},\frac{12}{21}\right]$ = [.476190, 0.571429]	[0.434369, 0.61325]
R	$\left[\frac{8}{21},\frac{12}{21}\right]$ = [0.380952, 0.571429]	[0.29731, 0.655071]
B|G	[0.3, 0.7]	[0.124352, 0.875649]
B|R	[0.06, 0.14]	[0.0248703, 0.17513]

Walley develops an imprecise beta-binomial model (IBB) as a special case of an imprecise Dirichlet model (IDM). We illustrate our indefinite probabilities approach as applied to Bayes' rule, under the same assumptions as these other approaches.

We treat here the standard form for Bayes' rule: $P(A|B) = \frac{P(A)P(B|A)}{P(B)}$.

We consider the following simple example problem. Suppose we have 100 gerbils of unknown color; 10 gerbils of known color, 5 of which are blue; and 100 rats of known color, 10 of which are blue. We wish to estimate the probability of a randomly chosen blue rodent being a gerbil.

The first step, in our approach, is to obtain initial probability intervals. We obtain the following sets of initial probabilities shown in Tables 1–3, corresponding to credibility levels *b* of 0.95, and 0.982593, respectively.

We begin our Monte Carlo step by generating n_1 random strength values, chosen from Beta distributions proportional to $x^{ks-1}(1-x)^{k(1-s)-1}$ with mean values of $s=\frac{11}{21}$ for *P*(G); $s=\frac{10}{21}$ for *P*(R); $s=\frac{1}{2}$ for $P(B|G)$; and $s=\frac{1}{10}$ for $P(B|R)$, and with support on [L1, U1]. Each of these strength values then serve, in turn, as parameters of standard

Table 3. Final Probability Intervals for P(G|B) using initial b-values of 0.90

CREDIBILITY LEVEL	INTERVAL
0.90	[0.715577, 0.911182]
0.95	[0.686269, 0.924651]

Table 4. Final Average Strength and Count Values using initial b-values of 0.90

CREDIBILITY LEVEL	STRENGTH	COUNT via n=k(1–w)/w
0.90	0.832429	41.1234
0.95	0.832429	31.9495

Table 5. Final Probability Intervals for P(G|B) using initial b-values of 0.95

CREDIBILITY LEVEL	INTERVAL
0.90	[0.754499, 0.896276]
0.95	[0.744368, 0.907436]

Table 6. Final Average Strength and Count Values using initial b-values of 0.95

CREDIBILITY LEVEL	STRENGTH	COUNT
0.90	0.835557	60.5334
0.95	0.835557	51.3239

Beta distributions. We generate a random sample of n_2 points from each of these standard Beta distributions.

We next apply Bayes' Theorem to each of of the $n_1 n_2$ quadruples of points, generating n_1 sets of sampled distributions. Averaging across each distribution then gives a distribution of final mean strength values. Finally, we transform our final distribution of mean strength values back to (*s, n, b*) triples.

5.1.1. Experimental Results

Results of our Bayes' Rule experiments are summarized in Tables 3–6.

5.1.2. Comparison to Standard Approaches

It is not hard to see, using the above simple test example as a guide, that our indefinite probabilities approach generalizes both classical Bayesian inference and Walley's IBB model. First note that single distributions can be modeled as envelopes of distributions with parameters chosen from uniform distributions. If we model *P*(B) as a uniform distribution; *P*(A) as a single beta distribution and $P(B|A)$ as a single binomial distribution, then our method reduces to usual Bayesian inference. If, on the other hand, we model *P*(B) as a uniform distribution; *P*(A) as an envelope of beta distributions; and $P(B|A)$ as an envelope of binomial distributions, then our envelope-approach reduces to Walley's IBB model. Our envelope-based thus approach allows us to model *P*(B) by any given family of distributions, rather than restricting us to a uniform distribution. This allows for more flexibility in accounting for known, as well as unknown, quantities.

To get a quantitative comparison of our approach with these others, we modeled the above test example using standard Bayesian inference as well as Walley's IBB model. To carry out standard Bayesian analysis, we note that given that there are 100 gerbils whose blueness has not been observed, we are dealing with 2^{100} "possible worlds" (i.e. possible assignments of blue/non-blue to each gerbil). Each of these possible worlds has 110 gerbils in it, at least 5 of which are blue, and at least 5 of which are non-blue.

For each possible world *w*, we can calculate the probability that drawing 10 gerbils from the population of 110 existing in world *W* yields an observation of 5 blue gerbils and 5 non-blue gerbils. This probability may be written $P(D|H)$, where *D* is the observed data (5 blue and 5 non-blue gerbils) and *H* is the hypothesis (the possible world *W*).

Applying Bayes' rule, we have $P(H|D)=\frac{P(D|H)P(H)}{P(D)}$. Assuming that $P(H)$ is constant across all possible worlds, we find that $P(H|D)$ is proportional to $P(D|H)$. Given this distribution for the possible values of the number of blue gerbils, one then obtains a distribution of possible values $P(gerbil|blue)$, and calculates a credible interval. The results of this Bayesian approach are summarized in Table 7.

We also applied Walley's IBB model to our example, obtaining (with k = 10) the interval $\left[\frac{2323}{2886},\frac{2453}{2886}\right]$, or approximately [0.43007, 0.88465]. In comparison, the hybrid method succeeds at maintaining narrower intervals, albeit at some loss of credibility.

With a k-value of 1, on the other hand, Walley's approach yields an interval of [0.727811, 0.804734]. This interval may seem surprising since it does not include the average given by Bayes' theorem. However, it is sensible given the logic of Walley's approach. In this approach, we assume no prior knowledge of P(G) and we have 10 new data points in support of the proposition that $P(G|B)$ is 55/65. So we assume beta distribution priors with s = 0 and s = 1 for the "endpoints" of P(G) and use n = 10 and p = 11/13 for the binomial distribution for $P(G|B)$.

The density function thus has the form:

Table 7. Final Probability Intervals for P(G|B)

CREDIBILITY LEVEL	**INTERVAL**
0.90	[0.8245614035, 0.8630136986]
0.95	[0.8214285714, 0.8648648649]

$$f(x)=\frac{x^{ks}(1-x)^{k(1-s)}x^{np}(1-x)^{n(1-p)}}{\int_0^1 x^{ks}(1-x)^{k(1-s)}x^{np}(1-x)^{n(1-p)}dx}$$

Now for s = 1 and value of the learning parameter k = 1, the system with no prior knowledge starts with the interval [1/3,2/3]. With only 10 data points in support of p = 11/13 and prior assumption of no knowledge or prior p = 1/2, Walley's method is (correctly according to its own logic) reluctant to move quickly in support of p = 11/13, without making larger intervals via larger k-values.

5.2. A Detailed Beta-Binominal Example for Deduction

Next we consider another inference rule, term logic deduction, which is more interesting than Bayes' Rule in that it combines probability theory with an heuristic independence assumption. The independence-assumption-based PLN deduction rule, as derived in ([13]), has the following form, for "consistent" sets *A*, *B*, and *C*[3]:

$$s_{AC}=s_{AB}s_{BC}+\frac{(1-s_{AB})(s_C-s_Bs_{BC})}{1-s_B}$$

where

$$s_{AC}=P(C|A)=\frac{P(A\cap C)}{P(A)}$$

assuming the given data $s_{AB}=P(B|A)$, $s_{BC}=P(C|B)$, $s_A=P(A)$, $s_B=P(B)$, and $s_C=P(C)$.

Our example for the deduction rule will consist of the following premise truth-values. In the table, we provide the values for [*L*, *U*], and *b*, as well as the values corresponding to the mean *s* and count *n*.

We now vary the premise truth-value for variable C, keeping the mean s_C constant, in order to study changes in the conclusion count as the premise width [L,U] varies. In the table below, *b* = 0.9 for both premise and conclusion, and $s_C = 0.59$. The final count n_{AC} is found via the heuristic formula $n_{AC} = k(1 - W)/W$.

[3] Consistency here is defined in terms of a set of inequalities interrelating the premise probabilities, which are equivalent to the requirement that the conclusion according to the formula given here must lie in the interval [0,1].

Table 8. Premise Truth-Values Used for Deduction with Symmetric Intervals

Premise	***s***	**[*L*,*U*]**	**[*L1*, *U1*]**
A	11/23	[10/23, 12/23] ≈ [0.434783, 0.521739]	[0.383226, 0.573295]
B	0.45	[0.44, 0.46]	[0.428142, 0.471858]
AB	0.413043	[0.313043, 0.513043]	[0.194464, 0.631623]
BC	8/15	[7/15, 9/15] ≈ [0.466666, 0.6]	[0.387614, 0.679053]

Table 9. Deduction Rule Results Using Symmetic Intervals

Premise *C*		**Conclusion *AC***		
[*L*,*U*]	**[*L1*, *U1*]**	**sAC**	**[*L*,*U*]**	***nAC***
[0.44, 0.74]	[0.262131, 0.917869]	0.575514	[0.434879, 0.68669]	24.0004
[0.49, 0.69]	[0.371421, 0.808579]	0.571527	[0.478815, 0.650717]	48.1727
[0.54, 0.64]	[0.48071, 0.69929]	0.571989	[0.52381, 0.612715]	102.478
[0.58, 0.60]	[0.568142, 0.611858]	0.571885	[0.4125, 0.6756]	197.892

For comparison of using symmetric intervals versus asymmetric, we also tried identical premises for the means using the asymmetric interval approach. In so doing, the premise intervals [*L*, *U*] and [*L1*, *U1*] are different as shown in the table below, using $b = 0.9$ as before.

Table 10. Premise Truth-Values Used for Deduction with Asymmetric Intervals

Premise	***s***	**[*L*,*U*]**	**[*L1*, *U1*]**
A	11/23	[0.44, 0.52]	[0.403229, 0.560114]
B	0.45	[0.44, 0.462222]	[0.42818, 0.476669]
AB	0.413043	[0.38, 0.46]	[0.3412, 0.515136]
BC	8/15	[0.48, 0.58]	[0.416848, 0.635258]

Table 11. Deduction Rule Results Using Symmetic Intervals

Premise *C*		**Conclusion *AC***		
[*L*,*U*]	**[*L1*, *U1*]**	**sAC**	**[*L*,*U*]**	***nAC***
[0.40, 0.722034]	[0.177061, 0.876957]	0.576418	[0.448325, 0.670547]	35
[0.45, 0.687288]	[0.28573, 0.801442]	0.577455	[0.461964, 0.661964]	40
[0.50, 0.652543]	[0.394397, 0.725928]	0.572711	[0.498333, 0.628203]	67
[0.55, 0.617796]	[0.526655, 0.600729]	0.568787	[0.4125, 0.6756]	125

5.2.1. Modus Ponens

Another important inference rule is Modus Ponens, which is the form of deduction standard in predicate logic rather than term logic. Term logic deduction as described above is preferable from an uncertain inference perspective, because it generally supports more certain conclusions. However, the indefinite probabilities approach can also handle Modus Ponens, it simply tends to assign conclusions fairly wide interval truth-values.

The general form of Modus Ponens is:

A
A → B
|-
B

To derive an inference formula for this rule, we reason as follows. Given that we know P(A) and P(B | A), we know nothing about P(B | ¬A). Hence P(B) lies in the interval [Q,R] = [P(A and B), 1 – (P(A) – P(A and B))] = [P(B|A) P(A), 1 – P(A) + P(B|A)P(A)].

For the Modus Ponens experiment reported here we used the following premises: For A, we used $([L,U],b) = \left(\left[\frac{10}{23},\frac{12}{23}\right],0.9\right)$; and for A → B, we used ([*L*,*U*],*b*) = ([0.313043,0.513043],0.9). We proceed as usual, choosing distributions of distributions for both P(A) and P(B | A). Combining these we find a distribution of distributions [Q,R] as defined above. Once again, by calculating means, we end up with a distribution of [Q,R] intervals. Finally, we find an interval [*L*,*U*] that contains (100⊕*b*)% of the final [Q,R] intervals. In our example, our final [*L*,*U*] interval at the *b* = 0.9 level is [0.181154, 0.736029].

5.3. Conjunction

Next, the AND rule in PLN uses a very simple heuristic probabilistic logic formula:

$$P(A \text{ AND } B)=P(A)P(B)$$

To exemplify this, we describe an experiment consisting of assuming a truth-value of ([*L*,*U*],b) = ([0.4,0.5],0.9) for A and a truth-value of ([*L*,*U*],b) = ([0.2,0.3],0.9).

The conclusion truth-value for *P(A* AND *B*) then becomes. ([*L*,*U*],b) = ([0,794882,0.123946],0.9).

5.4. Revision

The final inference rule we study in this paper, the "revision" rule, is used to combine different estimates of the truth-value of the same statement. Very sophisticated approaches to belief revision are possible within the indefinite probabilities approach; for instance we are currently exploring the possibility of integrating entropy optimization heuristics as described in ([14]) into PLN for this purpose ([15]). At present, however, we are using a relatively simple heuristic approach for revising truth-values. This approach seems to be effective in practice, although it lacks the theoretical motivation of the entropy minimization approach.

Suppose D1 is the second-order distribution for premise 1 and D2 is the second-order distribution for D2. Suppose further that n_1 is the count for premise 1 and n_2 is the count for premise 2. Let $w_1 = n_1/(n_1 + n_2)$ and $w_2 = n_2/(n_1 + n_2)$ and then form the conclusion distribution D = w_1 D1 + w_2D2. We then generate our $[L,U],b$ truth-value as usual.

As an example, consider the revision of the following two truth-values ([0.1, 0.2], 0.9) and ([0.3, 0.7], 0.9). Estimating the counts using $n = k1(1 - W)/W$ gives count values of 90 and 15 respectively. Fusing the two truth-values yields ([0.19136, 0.222344], 0.9) with a resulting count value of 79.1315.

6. Conclusions

Using credible intervals of probability distribution envelopes to model second-order plausibiltiies, we have generalized both Walley's imprecise probabilities model and the standard Bayesian model into a novel "indefinite probabilities" approach. On Bayes' rule, the results obtained via this method fall squarely between standard Bayesian models and Walley's interval model, providing more general results than Bayesian inference, while avoiding the quick degeneration to worst-case bounds inherent with imprecise probabilities. On more heuristic PLN inference rules, the indefinite probability approach gives plausible results for all cases attempted, as exemplified by the handful of examples presented in detail here.

Comparing the merits of the indefinite probability approach with those of other approaches such as Walley's imprecise probabilities, the standard Bayesian approach, or NARS is difficult, for conceptual rather than practical or mathematical reasons. Each of these approaches proceeds from subtly different theoretical assumptions. To a great extent, each one is the correct approach according to its own assumptions; and the question then becomes which assumptions are most useful in the context of pragmatic uncertain inferences. In other words: the ultimate justification of a method of quantifying uncertainty, from an AGI or narrow-AI perspective, lies in the quality of the inferential conclusions that can be obtained using it in practice.

However, the drawing of such conclusions is not a matter of uncertainty quantification alone – it requires an inference engine (such as PLN), embedded in a systematic framework for feeding appropriate data and problems to the inference engine (such as the Novamente AI Engine). And so, the assessment of indefinite probabilities from this perspective goes far beyond the scope of this paper. What we have done here is to present some initial, suggestive evidence that the indefinite probabilities approach may be useful for artificial general intelligence systems – firstly because it rests on a sound conceptual and semantic foundation; and secondly because when applied in the context of a variety of PLN inference rules (representing modes of inference hypothesized to be central to AGI), it consistently gives intuitively plausible results, rather than giving results that intuitively seem too indefinite (like the intervals obtained from Walley's approach, which too rapidly approach [0,1] after inference), or giving results that fail to account fully for premise uncertainty (which is the main issue with the standard, Bayesian or frequentist, first-order-probability approach).

References

[1] Pei Wang, “The Limitation of Bayesianism, Artificial Intelligence,” 158(1), 97–106, 2004.
[2] Peter Walley, “Statistical reasoning with imprecise probabilities,” London; New York: Chapman and Hall, 1991.
[3] John Maynard Keynes, A Treatise on Probability, Dover Books, 1921, 2004
[4] Kurt Weichselberger, Thomas Augustin: “On the Symbiosis of Two Concepts of Conditional Interval Probability,” ISIPTA 2003: 606.
[5] Peter Walley, “Inferences from Multinomial Data: Learning about a Bag of Marbles,” Journal of the Royal Statistical Society. Series B (Methodological), Vol. 58, No. 1, pp. 3–57, (1996).
[6] Stanford Encyclopedia of Philosophy, “Interpretations of Probability,” 2003, http://plato.stanford.edu/entries/probability-interpret/.
[7] Frank Ramsey, Foundations of Mathematics, 1931.
[8] Bruno de Finetti, Theory of Probability, (translation by AFM Smith of 1970 book) 2 volumes, New York: Wiley, 1974–5.
[9] R. T. Cox, “Probability, Frequency, and Reasonable Expectation,” Am. Jour. Phys., 14, 1–13, (1946).
[10] Michael Hardy, Advances in Applied Mathematics, August 2002, pages 243–292.
[11] Pei Wang, Rigid Flexibility: The Logic of Intelligence, Springer, 2006.
[12] Pei Wang, Non-Axiomatic Reasoning System: Exploring the essence of intelligence, Ph.D. Dissertation, Indiana University, 1995.
[13] Ben Goertzel, Matthew Iklé, Izabela Freire Goertzel, Ari Heljakka, Probabilistic Logic Networks: A General Framework for Applied Uncertain Inference, in preparation, expected publication 2007.
[14] Gabriele Kern-Isberner and Wilhelm Rödder. Belief revision and information fusion on optimum entropy, International Journal of Intelligent Systems, Volume 19, Issue 9, 2004, Pages 837–857.
[15] Ben Goertzel, Matthew Iklé, “Revision of Indefinite Probabilities via Entropy Minimization”, in preparation, expected publication 2007.

Advances in Artificial General Intelligence: Concepts, Architectures and Algorithms
B. Goertzel and P. Wang (Eds.)
IOS Press, 2007

© 2007 The authors and IOS Press. All rights reserved.

Virtual Easter Egg Hunting: A Thought-Experiment in Embodied Social Learning, Cognitive Process Integration, and the Dynamic Emergence of the Self

Ben GOERTZEL
Novamente LLC

Abstract. The Novamente Cognition Engine (NCE) architecture for Artificial General Intelligence is briefly reviewed, with a focus on exploring how the various cognitive processes involved in the architecture are intended to cooperate in carrying out moderately complex tasks involving controlling an agent embodied in the AGI-Sim 3D simulation world. A handful of previous conference papers have reviewed the overall architecture of the NCE, and discussed some accomplishments of the current, as yet incomplete version of the system; this paper is more speculative and focuses on the intended behaviors of the NCE once the implementation of all its major cognitive processes is complete. The "iterated Easter Egg Hunt" scenario is introduced and used as a running example throughout, due to its combination of perceptual, physical-action, social and self-modeling aspects. To aid in explaining the intended behavior of the NCE, a systematic typology of NCE cognitive processes is introduced. Cognitive processes are typologized as global, operational or focused; and, the focused processes are more specifically categorized as either forward-synthesis or backward-synthesis processes. The typical dynamics of focused cognition is then modeled as an ongoing oscillation between forward and backward synthesis processes, with critical emergent structures such as self and consciousness arising as attractors of this oscillatory dynamic. The emergence of models of self and others from this oscillatory dynamic is reviewed, along with other aspects of cognitive-process integration in the NCE, in the context of the iterated Easter Egg Hunt scenario.

Introduction

The Novamente Cognition Engine (NCE) is an in-development software system, ultimately aimed toward artificial general intelligence at the human level and beyond. It is programmed in C++ and designed for large-scale implementation on a distributed network of Linux machines. The overall architecture of the system is based on principles from cognitive science and complex systems science, and grounded in a systems theory of intelligence; but the specific mechanisms used within the system are drawn from contemporary computer science, utilizing on a number of recent advances in AI theory and practice. Among the key cognitive mechanisms of the Novamente system are a probabilistic reasoning engine based on a novel variant of probabilistic logic called Probabilistic Logic Networks; an evolutionary learning engine that is based on a synthesis of probabilistic modeling and evolutionary programming called MOSES, described in Moshe Looks' 2006 PhD thesis from Washington University (see [1]); and an artificial economics based system for attention allocation and credit assignment. Implementation of the system has been underway for some time but is not yet complete[1]; however, the current incomplete system has nevertheless been used to experiment with various forms of learning and reasoning. Among other applications, the NCE is now being used to control a simulated humanoid agent in 3D simulation world called AGISim (see [2]), similar to a video game world, and in this context is being taught simple behaviors such as playing fetch and tag, finding objects, and recognizing objects by name. These teaching exercises serve to give the system basic world-knowledge which can then be deployed beyond the scope of the simulation world; and they serve as the first stage of a principled process of "AGI developmental psychology" based loosely on the ideas of Jean Piaget ([3]).

The Novamente design has been overviewed in a number of conference papers before[2] ([4–7]) and our goal here is not to repeat this content, but rather to give a more elegantly conceptually organized perspective on the cognitive processes contained in the system, and to explore the means by which it is hypothesized these processes will be able to act together, in a coordinated way, to enable embodied social learning and to spawn the emergence within a Novamente system's dynamic knowledge base of a structure fairly describable as a "self" (or, to use a more technical term introduced by [8]), a "phenomenal self"). To get the most out of this paper, the reader should first read these prior overviews; but nonetheless the current paper has been written to be minimally self-contained.

Two recent papers ([2,9]) discuss learning and reasoning experiments that have been conducted with the current system; and, two other papers in this volume discuss aspects of the Novamente system as well. Looks' paper discusses the MOSES probabilistic evolutionary learning system created for incorporation into Novamente; and the Ikle' et al. paper discusses some of the mathematics underlying Novamente's Probabilistic Logic Networks (PLN), including a simple example of PLN inference applied to embodied learning in the AGISim simulation world. This paper is more speculative, and describes both implemented/tested and as-yet unimplemented/untested aspects of the NCE design, with a focus on some of the integrative cognitive dynamics and emergent structures that are expected to be observed once the implementation, testing and tuning of the system is more complete.

Section 2 gives a brief overview of the Novamente architecture, largely covering ground already reviewed in prior publications. Section 3 presents a typology of cognitive processes using the notions of forward and backward synthesis, and utilizes this to give a unified presentation of the various cognitive processes existing in the NCE, and to compose an hypothesis regarding the origins of the phenomenal self and the attentional focus from complex cognitive dynamics. Section 4 describes the iterated Easter Egg Hunt scenario, and discusses how the NCE's cognitive dynamics are expected to productively cooperate in this context, in a completely implemented NCE.

The system-theoretic focus makes this paper different than the typical paper in the AI literature; and this difference reflects the relatively unusual nature of the system-theoretic methodology underlying the NCE project. In our view, the only kind of approach to AGI likely to meet with success is one in which the systematic coordinated

[1] The implementation process is time-consuming not only due to the usual difficulties of large-scale software engineering, but also due to the need to carefully define and test a large number of details not fully specified in the overall AI design.

behavior of a holistic AGI system is carefully envisioned prior to the detailed implementation and testing of the system. I.e., I suggest that the parts of an AGI system must be designed, in detail, with a specific view toward causing the whole to behave appropriately. Consistently with this, I assign low success odds to "integrative" designs in which multiple "best of breed" narrow-AI or proto-AGI components created independently by multiple research teams are hooked together within a flexible overall design. And, I assign a fundamental necessity to in-depth speculative exercises such as the one reported in Section 4 here – if AGI is to be achieved via computer science methods (rather than, for example, by detailed emulation of the human brain).

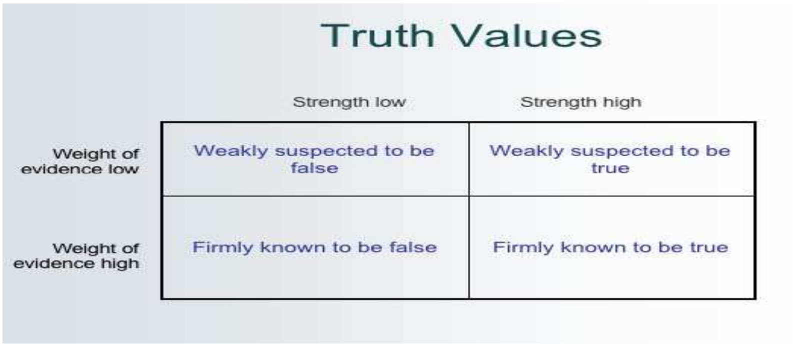
Above it was stated that the ultimate objective of NCE development is to create artificial intelligence at the human level and beyond. More specifically, the long-term objective of NCE development is to create a powerful "artificial scientist" that can ingest a vast variety of quantitative, relational and textual data from the Internet and then interact with human scientists to produce innovative scientific creations and discoveries. It may seem that Easter Egg hunting – which in its more sophisticated social-learning aspects is still beyond the current Novamente implementation – is an extremely long way from this sort of ambitious goal. But according to the developmental-psychology paradigm outlined in ([9]), there is a very natural path from these early-childhood-type learning tasks to the more complex and formal learning tasks required for doing science. The hypothesis motivating our current work on early-childhood-level learning tasks is that doing science does not require any cognitive mechanisms, knowledge representations or architecture components beyond those required for doing Easter Egg hunting (or other similar tasks) in a robust, adaptive, socially and reflectively aware way.

1. Brief Overview of the Novamente Architecture

One may describe the Novamente AGI architecture in terms of four different aspects: knoweldge representation, cognitive architecture, cognitive processes, and emergent structures. This section deals mainly with the former two; and the next section deals mainly with the latter two.

1.1. NCE Knowledge Representation

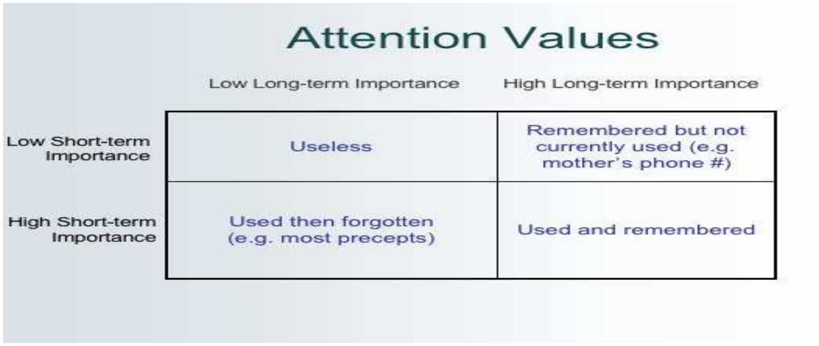
The NCE utilizes a knowledge representation in which declarative knowledge is represented using weighted, labeled hypergraphs; procedural knowledge is represented using programs in a customized functional programming language called Combo; and mechanisms are in place for freely converting between declarative and procedural knowledge. Nodes and links in the declarative-knowledge hypergraph are grouped together into the category of "Atoms." Atoms are quantified with truth values (see Fig. 1) that, in their simplest form, have two components, one representing probability ("strength") and the other representing "weight of evidence"; and also with "attention values" (see Fig. 2) that have two components, short-term and long-term importance, representing the estimated value of the Atom on immediate and long-term time-scales.



Truth Values

	Strength low	Strength high
Weight of evidence low	Weakly suspected to be false	Weakly suspected to be true
Weight of evidence high	Firmly known to be false	Firmly known to be true

Figure 1. A coarse-grained view of the semantics of the truth values attached to Novamente Atoms.



Attention Values

	Low Long-term Importance	High Long-term Importance
Low Short-term Importance	Useless	Remembered but not currently used (e.g. mother's phone #)
High Short-term Importance	Used then forgotten (e.g. most precepts)	Used and remembered

Figure 2. A coarse-grained view of the semantics of the attention values attached to Novamente Atoms.

The vocabulary of node and link types used to represent knowledge in Novamente has been presented in ([4]). Figures 3 and 4 give some simple examples. An incomplete list of node types is as follows:

- ConceptNodes
  - o "tokens" for links to attach to
- PredicateNodes
- ProcedureNodes
- PerceptNodes
  - o Visual, acoustic percepts, etc.
- NumberNodes

And an incomplete list of link types is:

- Logical links
  - o InheritanceLink
  - o SimilarityLink
  - o ImplicationLink
  - o EquivalenceLink
  - o Intensional logical relationships
- HebbianLinks
- Procedure evaluation links

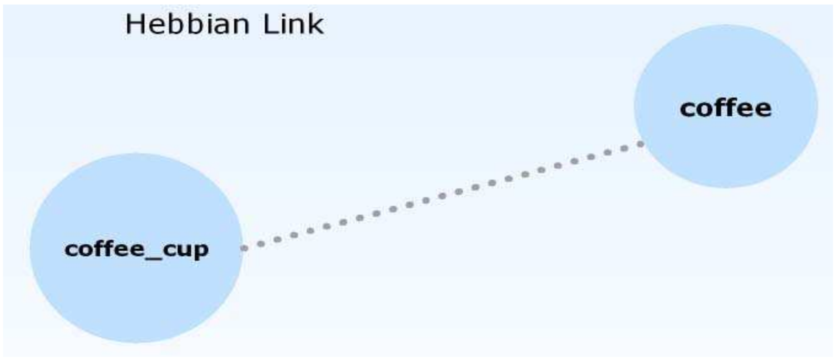


Figure 3. HebbianLinks may denote generic association, calculated based on the degree to which two Atoms have proved simultaneously useful to the system.

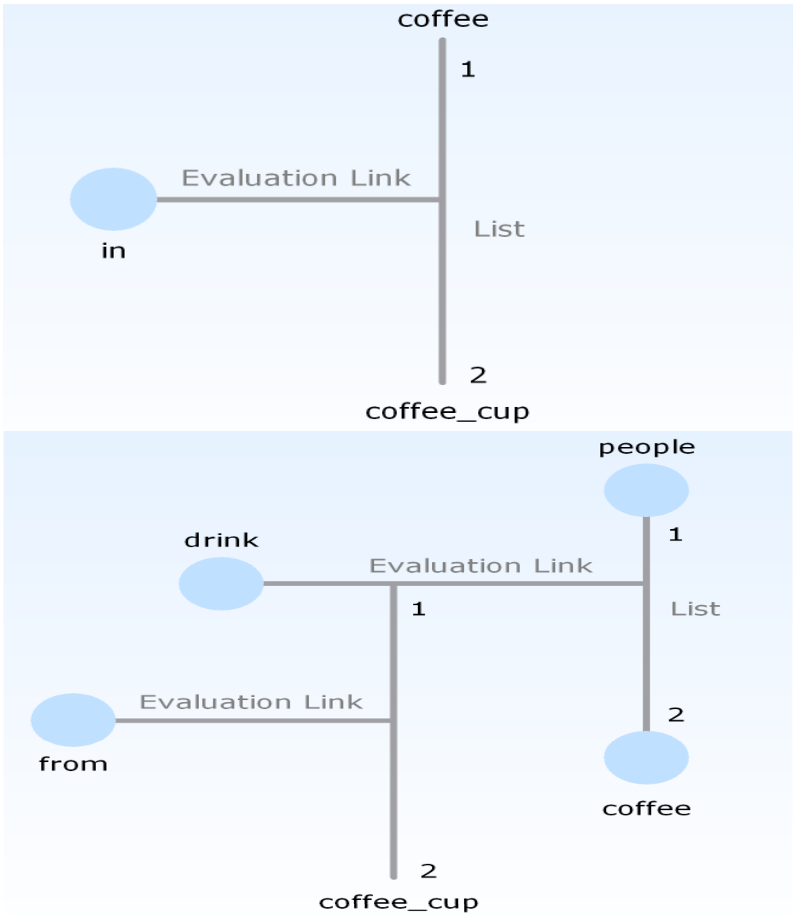


Figure 4. Links may also denote precise logical relationships, quantified with degrees of uncertainty and importance (quantifications not shown in the diagram).

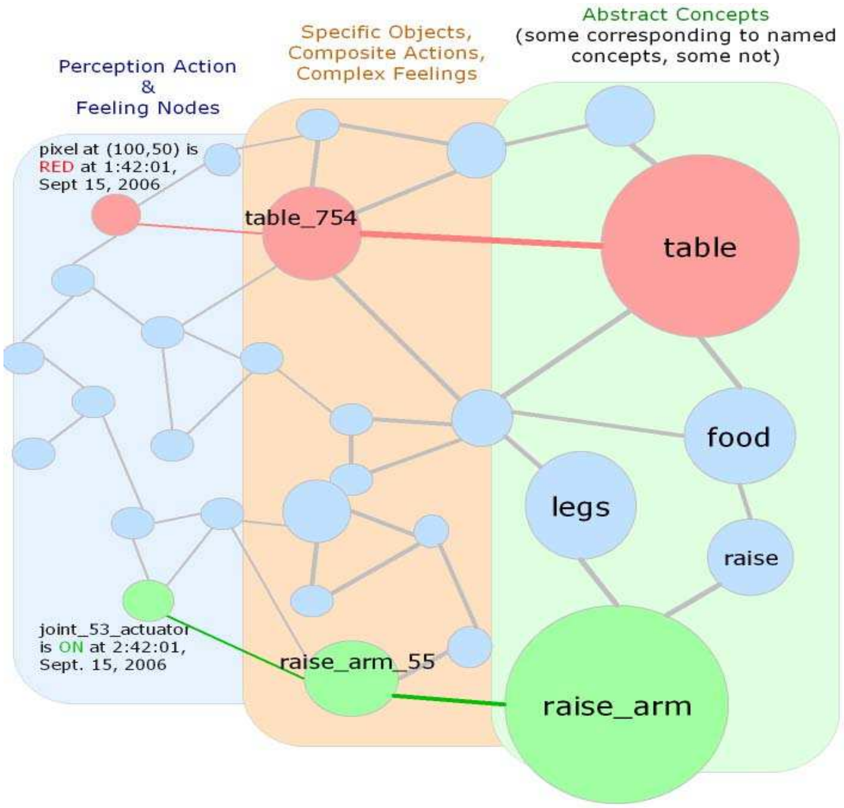

Abstract Concepts
(some corresponding to named concepts, some not)
Specific Objects, Composite Actions, Complex Feelings
Perception Action & Feeling Nodes
pixel at (100,50) is RED at 1:42:01, Sept 15, 2006
table_754
table
food
legs
raise
joint_53_actuator is ON at 2:42:01, Sept. 15, 2006
raise_arm_55
raise_arm

Figure 5. Novamente Atoms may refer to a variety of scales of specificity, from specific percepts or action-commands, to specific entities (like one particular table or one particular action of raising the arm), to general concepts. Some nodes in the diagram are labeled and some are not, reflecting the fact that in the actual Novamente Atomspace many nodes do not correspond to any particular English word nor any concept or entity compactly describable in English.

The semantics of these types is discussed in prior publications.

It is important to note that the Novamente declarative knowledge representation is neither a neural net nor a semantic net, though it does have some commonalities with each of these traditional representations. It is not a neural net because it has no activation values, and involves no attempts at low-level brain modeling. However, "attention values" are very loosely analogous to time-averages of neural net activations.On the other hand, it is not a semantic net because of the broad scope of the Atoms in the network (see Fig. 5): for example, Atoms may represent percepts, procedures, or parts of concepts. Most Novamente Atoms have no corresponding English label. However, most Novamente Atoms do have probabilistic truth values, allowing logical semantics.

1.2. NCE Cognitive and Software Architecture

The overall NCE software architecture consists of a collection of specialized units, each of which uses the same knowledge representations and cognitive mechanisms to achieve a particular aspect of intelligence, such as perception, language learning, abstract cognition, action selection, procedure learning, etc. The breakdown into units,

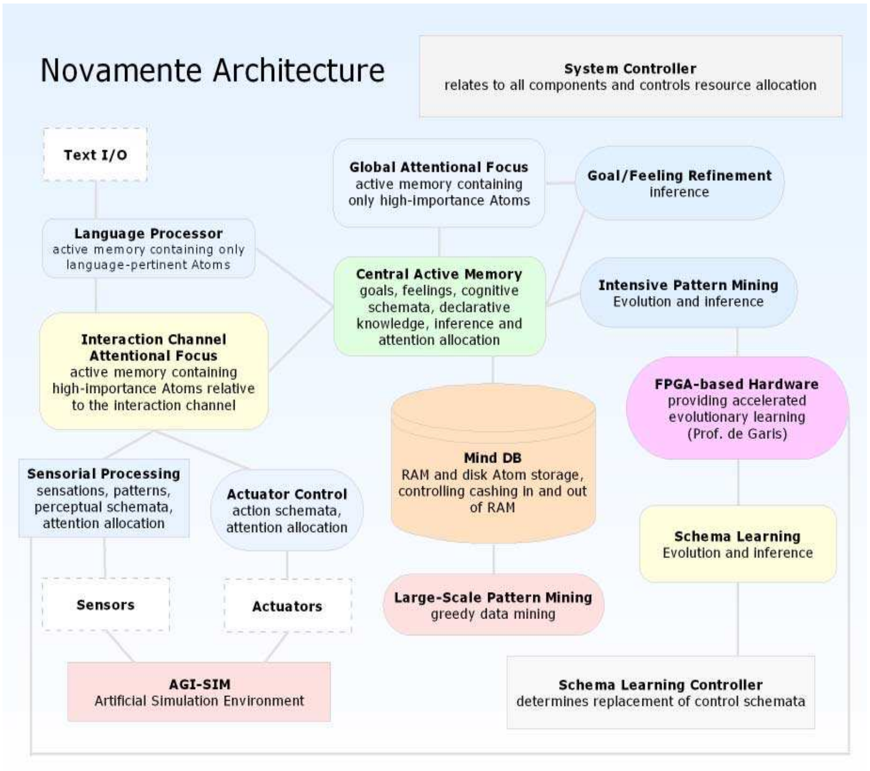

Novamente Architecture
System Controller
relates to all components and controls resource allocation
Text I/O
Global Attentional Focus
active memory containing only high-importance Atoms
Goal/Feeling Refinement
inference
Language Processor
active memory containing only language-pertinent Atoms
Central Active Memory
goals, feelings, cognitive schemata, declarative knowledge, inference and attention allocation
Intensive Pattern Mining
Evolution and inference
Interaction Channel Attentional Focus
active memory containing high-importance Atoms relative to the interaction channel
FPGA-based Hardware
providing accelerated evolutionary learning (Prof. de Garis)
Mind DB
RAM and disk Atom storage, controlling cashing in and out of RAM
Sensorial Processing
sensations, patterns, perceptual schemata, attention allocation
Actuator Control
action schemata, attention allocation
Schema Learning
Evolution and inference
Sensors
Actuators
Large-Scale Pattern Mining
greedy data mining
AGI-SIM
Artificial Simulation Environment
Schema Learning Controller
determines replacement of control schemata

Figure 6. A high-level cognitive architecture for Novamente, in which most of the boxes correspond to Units in the sense of Fig. 7. This diagram is drawn from (XX).

indicated in Fig. 8, is based loosely on ideas from cognitive science, and is fairly similar to that presented in other integrative AGI architectures proposed by [11,12] and others. However, the specifics of the breakdown have been chosen with care, with a view toward ensuring that the coordinated dynamics of the mechanisms and units will be able to give rise to the emergent structures and dynamics associated with intelligence, including a sophisticated self-model and an appropriately adaptive "moving focus of attention."

As shown in Fig. 7, each individual unit within the NCE architecture contains a table of Atoms and Combo trees representing knowledge, and then a collection of processes called MindAgents that collaboratively act on the AtomTable. The Mind Agents embody various cognitive processes, as will be described below. A unit may span several machines or may be localized on a single machine: in the multi-machine case, Nodes on one machine may link to Nodes living in other machines within the same unit. On the other hand, Atoms in one Unit may not directly link to Atoms in another Unit; though different Units may of course transport Atoms amongst each other. This architecture is workable to the extent that Units may be defined corresponding to pragmatically distinct areas of mental function, e.g. a Unit for language processing, a Unit for visual perception, etc.

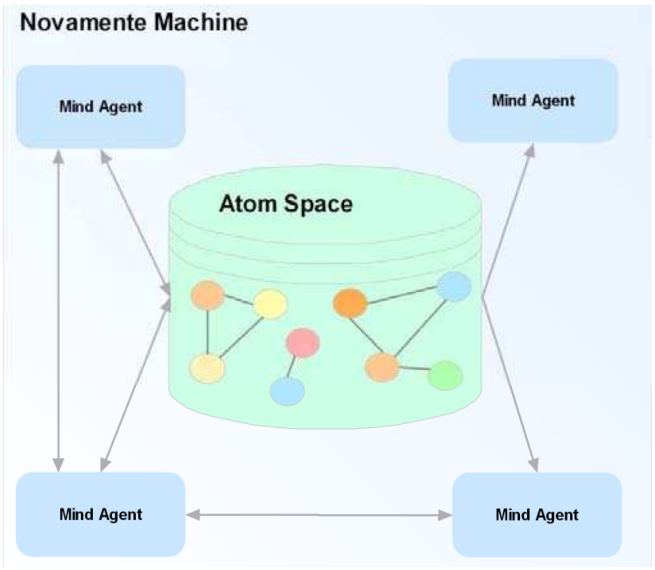


Figure 7. The architecture of a single Novamente "lobe", involving a table of Atoms and then a collection of processes called MindAgents that collaboratively act on the AtomTable. A Lobe may run on one machine or several. A Scheduler object regulates the activity of the MindAgents; and the MindAgents may spawn objects called Tasks that carry out one-time cognitive actions, and are scheduled using a ticketing system. Some MindAgents carry out actions not centered on the AtomTable, such as the SchemaExecution MindAgent which executes procedures in the manner of a programming language interpreter.

Next, Fig. 9 gives a simplified, schematic view of the flow of information through the Novamente system, in the context of its use to control an agent in the AGISim simulation world. This diagram does not cover "background reasoning and learning" processes that serve to create knowledge useful in figuring out how to achieve goals.

The full cognitive architecture has not yet been implemented; the currently implemented subset of the architecture will be briefly reviewed at the start of Section 4.

2. A Typology of Cognitive Processes in Novamente

The most complex aspect of the NCE design is the set of cognitive processes involved. This set of processes may be presented in a number of different ways. One approach is to look at the set of fundamental "AI learning" algorithms that are involved. In this sense, there are four main algorithmic approaches (each one of which involves a combination of a number of specific algorithms):

- Probabilistic Term Logic, a novel approach to inference under uncertainty, which is partially described in ([13]) in this volume.
- Probabilistic Evolutionary Learning, which in the current Novamente version takes the form of the MOSES algorithm, discussed in ([14]) in this volume.
- Stochastic pattern mining, which finds combinations of Atoms that satisfy a specified criterion (such as frequency of occurrence, statistical surprisingness, etc.)

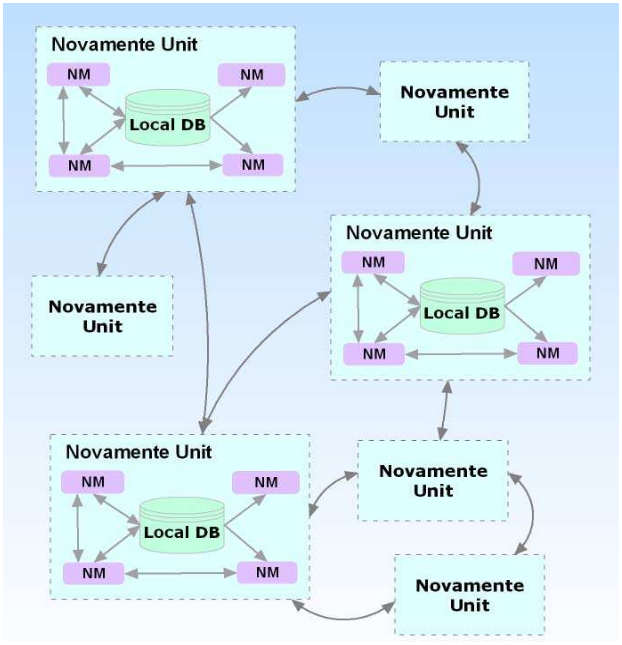


Figure 8. The high-level architecture of Novamente consists of a set of Units, each of which consists of a set of machines embodying the basic one-machine Novamente architecture. All the machines in a Unit share a common Atomspace, i.e. links in the AtomTable of one machine in a Unit may point to Atoms in another machine in the Unit. On the other hand, different Units have separate Atomspaces and Atoms in one Unit may not directly link to Atoms in another Unit; though different Units may of course transport Atoms amongst each other. This architecture is workable to the extent that Units may be defined corresponding to pragmatically distinct areas of mental function, e.g. a Unit for language processing, a Unit for visual perception, etc.

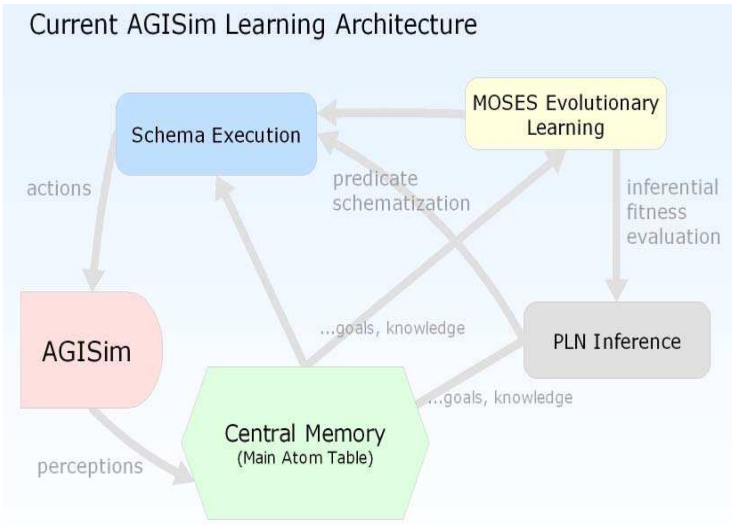


Figure 9. Schematic diagram of the architecture of the current NAIE implementation, which only implements part of the overall design, but successfully carried out various simple learning tasks in the AGISim environment.

- Artificial economics, used for attention allocation and credit assignment, leading to novel forms of adaptively evolving distributed knowledge representation.

In previous review papers, the set of cognitive processes involved in the Novamente architecture has been presented in a somewhat unorganized manner, as a simple list of MindAgents. Such a presentation is not inaccurate, but gives the impression of even more conceptual complication than actually exists in the design. In this section I will present a typology of Novamente cognitive processes, which makes clear that the diverse population of MindAgents in Novamente naturally subdivides into a handful of high-level conceptual categories. Furthermore (though this has not been validated yet), I conjecture that this typologizing exercise may have conceptual value beyond the scope of Novamente, by imposing a natural ontology on the diversity of cognitive processes necessarily present in any complex, multifaceted AI system.

First of all, at the highest level, I divide the cognitive processes occurring in Novamente into three categories:

- Global processes
  - MindAgents that periodically iterate through all Atoms and act on them
  - "Things that all Atoms do"
- Control Processes
  - Processes directly involved with carrying out specifically orchestrated sequences of actions in the external world or in the system's own infrastructure
- Focused processes
  - MindAgents that begin by selecting a small set of important or relevant Atoms, and then act on these to generate a few more small sets of Atoms, and iterate.

I suggest that processes in all these categories are critical to Novamente's intelligence (and more hypothetically, to intelligence in general). However, we also suggest that the greatest subtlety lies in the focused cognitive processes. The global and control processes are extremely necessary, however, my conjecture is that these processes don't necessarily need to be a lot more sophisticated in a human-level intelligence than in a significantly sub-human-level intelligence. The distinguishing characteristic of human-level intelligence, I suggest, is a much more highly-developed set of focused cognitive processes. To better understand focused cognitive processes in a general way, I will utilize the ideas of forward and backward synthesis developed in ([15]).

Key examples of Novamente control processes are as follows:

- Schema Execution
  - This involves the "programming language interpreter" used to actually execute schemata created from NM Atoms
- Maintenance of "active schema pool" (SchemaSelection MindAgent)
  - This involves choosing which procedures to place in the pool of "currently executing schemata" (meaning, schemata that are currently ready to be activated if their input conditions are met)

- Maintenance of "active goal pool" (FeasibilityUpdating MindAgent)
    - Determination of the set of predicates that are currently actively considered as system goals, which is done by updating "feasibility" information regarding the achievability of various goals given various committments of resources

These are processes that are necessary in order for the system to carry out actions; including the actual execution of actions, the selection of subgoals to be used to determine which actions to carry out, the choice of which actions to carry out, and various specialized actions such as the updating of system control schemata based on meta-learning. In a sense these are "focused cognitive processes" but they have a different nature than the processes to be analyzed here under that label: they are not concerned with taking a set of knowledge items and focused-ly learning more about it, but are, rather, concerned with actually doing things either in the external world or in the system's own infrastructure.

Next, key examples of global processes are

- Attention Allocation
    - Updates short and long term importance values associated with Atoms
    - Uses a "simulated economy" approach, with separate currencies for short and long term importance
- Stochastic pattern mining of the AtomTable
    - A powerful technique for predicate formation
    - Critical for perceptual pattern recognition as well as cognition
    - Pattern mining of inference histories critical to advanced inference control
- Building of the SystemActivityTable
    - Records which MindAgents acted on which Atoms at which times
    - Table is used for building HebbianLinks, which are used in attention allocation

Finally, focused cognitive processes may be decomposed into forward versus backward synthesis processes. The distinction between forward and backward synthesis processes is drawn in detail in ([15]), and depicted in Figs 10 and 11. Roughly, forward synthesis begins with a small set of entities, and then combines them (with each other and with other entities) to form new entities, and iterates this process. On the other hand, backward synthesis begins with a small set of entities, and tries to figure out how to generate this set via forward synthesis starting from some other set of entities that meets certain criteria (such as, in the case of backward inference, being confidently known).

Novamente's key forward synthesis processes are as follows:

- Forward-Chaining Probabilistic Inference (see Fig. 12)
    - Given a set of knowledge items, figure out what (definitely or speculatively) follows from it according to the rules of probabilistic term logic
- Concept/Goal Formation (see Fig. 13)
    - "Blend" existing concepts or goals to form new ones
    - [16] have explored the cognitive significance of this sort of blending operation in great detail.

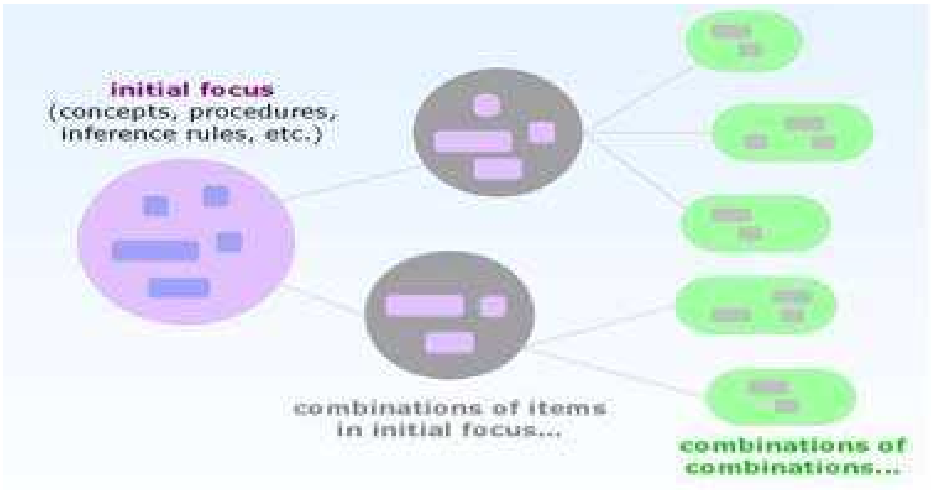

initial focus
(concepts, procedures, inference rules, etc.)
combinations of items in initial focus...
combinations of combinations...

Figure 10. The general process of forward synthesis.

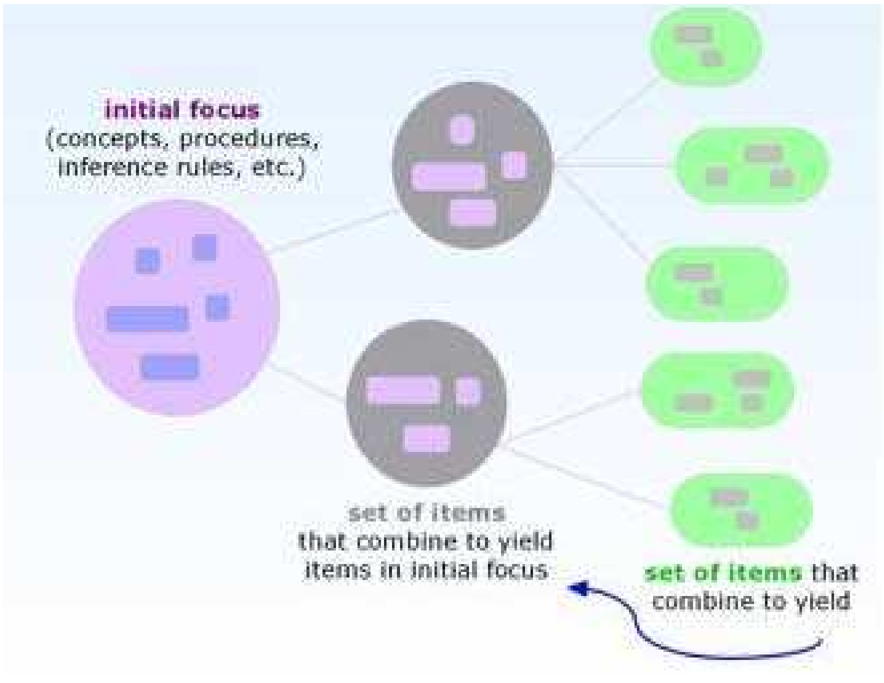

initial focus
(concepts, procedures, inference rules, etc.)
set of items that combine to yield items in initial focus
set of items that combine to yield

Figure 11. The general process of backward synthesis.

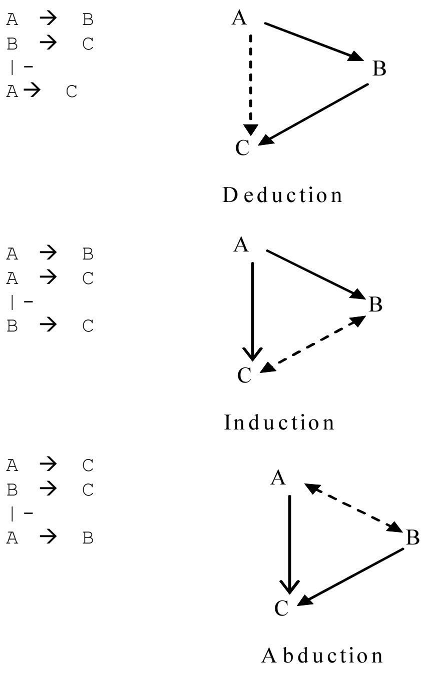


Figure 12. Probabilistic Logic Networks for uncertain inference: an illustration of first-order deduction, induction and abduction acting on extensional InheritanceLinks. This is a forward-synthesis process which chooses pairs of links and combines them to form new links, with truth values determined by probabilistic inference rules.

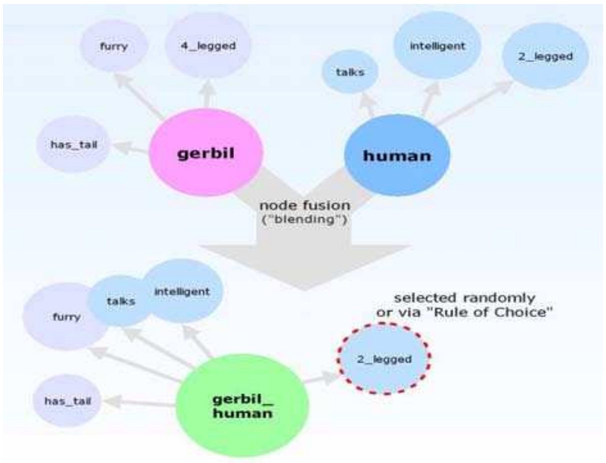


Figure 13. One of Novamente's heuristics for new-concept creation is "blending," in which some links from one concept are combined with some links from another concept.

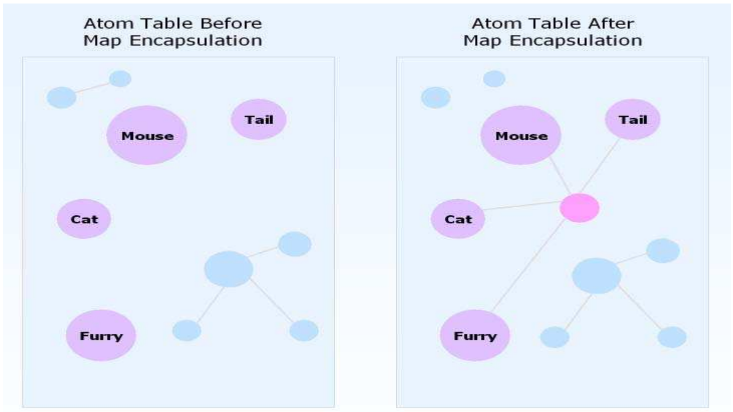

Atom Table Before Map Encapsulation
Atom Table After Map Encapsulation
Mouse
Tail
Cat
Furry
Mouse
Tail
Cat
Furry

Figure 14. Atoms commonly used together may be grouped together via linking them all to a newly created Atom. This process is called “map formation” and is one way that the Novamente system can effectively recognize patterns in itself.

- Map formation (see Fig. 14)
  - Create new Atoms out of sets of Atoms that tend to be simultaneously important (or whose importance tends to be coordinated according to some other temporal pattern)
- Language Generation
  - A subcategory of forward-chaining inference, but important enough to be isolated and considered on its own
  - Atoms representing semantic relationships are combined with Atoms representing linguistic mapping rules to produce Atoms representing syntactic relationships, which are then transformed into sentences
- Importance Propagation
  - Atoms pass some of their “attentional currency” to Atoms that they estimate may help them become important again in the future

Next, Novamente’s key backward synthesis processeses are:

- Backward-chaining probabilistic inference
  - Given a target Atom, find ways to produce and evaluate it logically from other knowledge
- Inference process adaptation
  - Given a set of inferential conclusions, find ways to produce those conclusions more effectively than was done before
- Predicate Schematization (Fig. 15)
  - Given a goal, and knowledge about how to achieve the goal, synthesize a procedure for achieving the goal
- Credit Assignment
  - Given a goal, figure out which procedures’ execution, and which Atoms’ importance, can be expected to lead to the goal’s achievement
- Goal Refinement
  - Given a goal, find other (sub)goals that imply that goal

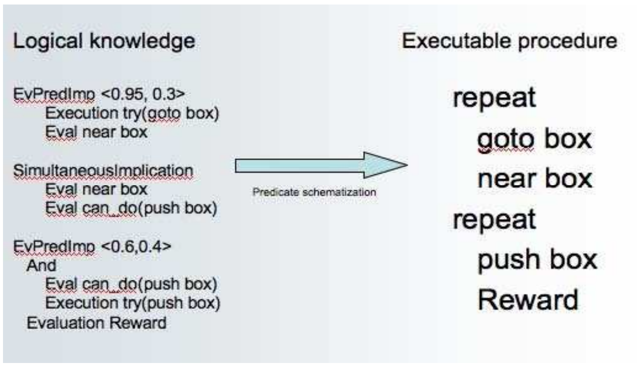


Figure 15. Predicate schematization: the backward synthesis process that maps declarative knowledge about how to achieve goals, into procedural knowledge that may be executed to actually achieve goals.

- Model-Based Predicate Generation
  - Given probabilistic knowledge about what patterns characterize predicates or procedures satisfying a certain criterion, generate new predicate/procedures satisfying the criterion
- Criterion-Based Predicate Modeling
  - Building of probabilistic knowledge regarding the patterns characterizing predicates satisfying a certain criterion
- Language Comprehension
  - Syntax parsing: given a sentence, or other utterance, search for assignments of syntactic relationships to words that will make the sentence grammatical
  - Semantic mapping: Search for assignment of semantic meanings to words and syntactic relationships that will make the sentence contextually meaningful

As a simple example of forward and backward synthesis in action, observe the example of embodied learning of commonsense knowledge shown in Fig. 16, reflecting actual experiments conducted with the current version of the NCE. In these experiments, MOSES was used to ground the concept of "nearness" in a simulation world. MOSES was able to learn the rule that nearness is transitive – via "backward synthesis," i.e. the search for rules simplifying and explaining the available data. Then, through use of the RelEx English language parsing front end, simple natural language sentences regarding the nearness of different countries to each other were entered in, and translated into semantic nodes and links in the NCE. (This is not the way a mature NCE AGI would process language, but it is a useful approach for preliminary experimentation while the NCE is not yet at the stage where it can learn English language on its own via its embodied social experience.) The PLN inference system can then apply the learned transitivity of nearness to the knowledge gained via language processing – a simple example of forward-chaining inference, resulting in the conclusion that (in the example in the Figure) Korea is slightly near Pakistan. The general process exemplified in this Figure is an important one. Embodied experience is learned to gain commonsense knowledge, which is expressed in abstract form, and can then be inferentially applied to domains remote from the one in which the knowledge was originally gained.

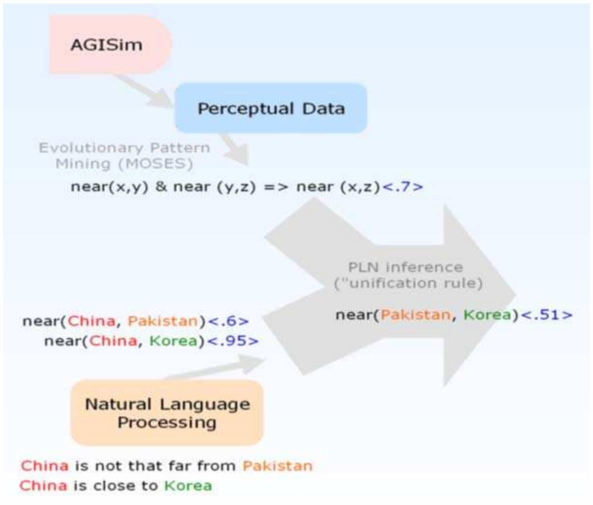
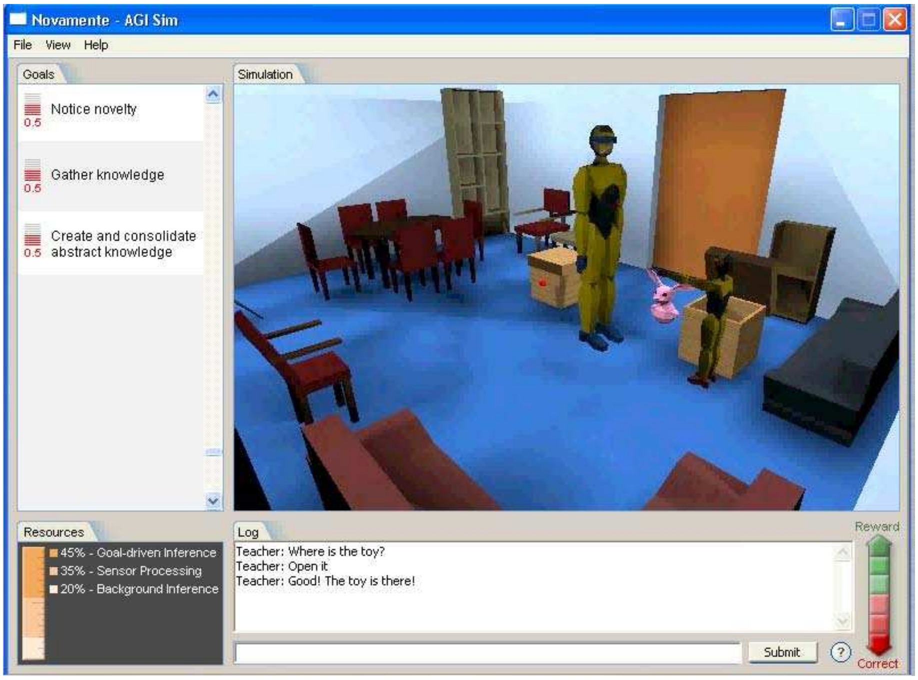

AGISim
Perceptual Data
Evolutionary Pattern Mining (MOSES)
near(x,y) & near (y,z) => near (x,z)<.7>
PLN inference ("unification rule)
near(Pakistan, Korea)<.51>
near(China, Pakistan)<.6>
near(China, Korea)<.95>
Natural Language Processing
China is not that far from Pakistan
China is close to Korea

Figure 16. A simple illustration of symbol grounding achieved through integrative intelligence. Evolutionary pattern mining discovers the grounding of the word "near" in perceived (AGISim) relationships, and then discovers the transitivity of nearness through analysis of multiple examples of perceived nearness. Language processing understands that China and Korea are near each other, and that China and Pakistan are somewhat near each other. Inference may then combine these facts learned by language processing with the transitivity of nearness learned via evolutionary learning on percepts gained via embodied experience, and conclude that Pakistan is somewhat near Korea. While simple, this is a nontrivial example of grounding in the sense that involves the grounding of an abstract relationship (transitivity of nearness) in perceived reality and then transferral of this relationship to nonperceived reality.

3. Embodied Social Cognition in the AGISim Environment

Now we briefly move from cognitive processes to another critical topic: methodologies of instruction. There is a variety of methods by which an AGI system may viably gain knowledge, including but not limited to:

- physically embodied experience, via robotic embodiment
- virtually embodied experience, via embodiment within a simulation world
  - example: the AGISim world, shown in Fig. 17
- non-embodied experience
  - e.g. via interaction with various online software agents
- conversation with humans
- reading in structured data from databases
  - databases of general knowledge constructed for other purposes
  - relational databases
  - the Mizar mathematics database
  - quantitative scientific, financial, etc. data
  - knowledge DB's constructed specifically for AI's and other software programs
    - everyday knowledge oriented DB's, e.g. Cyc [17]
    - linguistics oriented DB's, e.g WordNet [18], FrameNet [19]



Teacher: Where is the toy?
Teacher: Open it
Teacher: Good! The toy is there!

Figure 17. The Piagetan "A not B" problem presented to Novamente (the small humanoid agent) in the AGISim simulation world.

- reading knowledge encoded in language
    - natural language texts, e.g. online texts or textbooks
    - texts written for AI's in constructed languages like Lojban or Lojban++ [20]

One of the methodological principles underlying the Novamente approach to AGI education is that there is no need to choose between these. Given the right AI design you can have your cake and eat it too. Initially, the NCE is being instructed via a combination of database-ingestion and virtually embodied experience. In the next phase, we plan to augment this with a combination of instruction in English and in Lojban++. This flexibility is enabled because the NCE's knowledge representation permits explicit representation of knowledge (e.g. the creation of nodes corresponding to English-language concepts), but also permits experiential learning and implicit representation of knowledge in terms of learned patterns. The intention is that once the NCE becomes clever enough it will learn mainly via reading knowledge encoded in language, and conversational interaction with humans. Embodiment in physical robots may also be interesting but is not viewed as critically necessary.

Our initial goal in teaching the NCE in the AGISim world is simply to make the system able to learn infant-level behaviors "without cheating" – i.e. with the only instruction being interactions with a human-controlled agent in the simulation world. Example behaviors desired here are: naming objects, asking for objects, fetching objects, finding hidden objects, playing tag. The system will be tested using a set of tasks derived from human developmental psychology, a process that is already ongoing with tasks such as word-object association and Piaget's A-not-B task ([3,21]); see Fig. 17

for a depiction of this problem as presented to a Novamente-controlled agent in the AGISim simulation world).

The next step beyond this has to do with language understanding. Via instructing the system in language simultaneously with interacting with it in the simulation world, we believe that the system will be taught language in a way that it really understands it pragmatically and personally, unlike the kind of quasi-understanding possessed by current statistical or rule-based natural language systems. Once it is partway through this stage, it will possess the ability to learn from human teachers via linguistic communication utilizing complex recursive phrase structure grammar and grounded semantics. After this point is reached, we anticipate that future progress will accelerate considerably.[3]

Tied up with language understanding, of course, are social interaction and self-understanding, as argued by [22]. If all goes as envisioned, then as NCE improves its communicative ability, it will also improve its self-understanding. This process may be understood in terms of the concepts of forward and backward synthesis introduced above, as will be discussed a little later.

3.1. The Currently Implemented NCE/AGISim Architecture

Currently, as we work toward making a more and more completely functional and robust "artificial baby," we are working with a partial version of the NCE as depicted in Fig. 9 above, incorporating the following components:

- **Novamente core system**
  - AtomTable, MindAgents, Scheduler, etc.
  - Now runs on one machine; designed for distributed processing
- **PLN**
  - Relatively crude inference control heuristics
  - Simplistic predicate schematization
- **MOSES**
  - Little experimentation has been done evolving procedures with complex control structures
  - Not yet fully integrated with PLN
- **Schema execution framework**
  - Enacts learned procedures
- **AGISim**
  - And proxy for communication with NM core
- **Perception**
  - Stochastic conjunctive pattern mining for finding repeated patterns in data coming in from AGISim
- **NLP front end**
  - External NLP system for "cheating" style knowledge ingestion in the form of logical predicates, without the AGI system itself understanding the syntactic rules

[3] Also, at this stage, the symbol groundings learned by the system will be valuable for various narrow-AI purposes, such as natural language question answering.

Using this restricted system, we are working with simple tasks such as fetch, tag, word-object association and the Piagetan A-not-B experiment. The current version has proved capable of carrying out these tasks in various cases, and we conjecture that it will be capable of robustly carrying out a variety of similar tasks. However, to move beyond the infantile level we will clearly need to implement more of the NCE architecture, including most critically the economic attention allocation component; and we will need to integrate the MOSES and PLN components more fully than has been done so far.

3.2. The Emergence of the Self via Embodied Social Learning

As noted above, one of the key things we hope to see via teaching the NCE in the AGISim environment is the adaptive emergence within the NCE's knowledge base of an active and effectively evolving "phenomenal self." The process of the emergence of the self may, we hypothesize, be productively modeled in terms of the processes of forward and backward synthesis discussed above. This point is made carefully in [14] and just briefly summarized here.

What is ventured there is that the dynamic pattern of alternating forward and backward synthesis may play a fundamental role in cognition. Put simply, forward synthesis creates new mental forms by combining existing ones. Then, backward synthesis seeks simple explanations for the forms in the mind, including the newly created ones; and, this explanation itself then comprises additional new forms in the mind, to be used as fodder for the next round of forward synthesis. Or, to put it yet more simply:

… Combine … Explain … Combine … Explain … Combine …

This sort of dynamic may be expressed formally, in a Novamente context, as a dynamical iteration on the space of Atoms. One may then speak about attractors of this iteration: fixed points, limit cycles and strange attractors. And one may hypothesize some key emergent cognitive structures are strange attractors of this equation. I.e., the iterative dynamic of combination and explanation leads to the emergence of certain complex structures that are, in essence, maintained when one recombines their parts and then seeks to explain the recombinations. These structures are built in the first place through iterative recombination and explanation, and then survive in the mind because they are conserved by this process. They then ongoingly guide the construction and destruction of various other temporary mental structures that are not so conserved. Specifically, we suggest that both self and attentional focus may be viewed as strange attractors of this iteration. Here we will focus only on self.

The "self" in the present context refers to the "phenomenal self" [8] or "self-model." That is, the self is the model that a system builds internally, reflecting the patterns observed in the (external and internal) world that directly pertain to the system itself. As is well known in everyday human life, self-models need not be completely accurate to be useful; and in the presence of certain psychological factors, a more accurate self-model may not necessarily be advantageous. But a self-model that is too badly inaccurate will lead to a badly-functioning system that is unable to effectively act toward the achievement of its own goals.

The value of a self-model for any intelligent system carrying out embodied agentive cognition is obvious. And beyond this, another primary use of the self is as a foundation for metaphors and analogies in various domains. A self-model can in many cases form a self-fulfilling prophecy (to make an obvious double-entendre'!). Actions are generated based on one's model of what sorts of actions one can and/or should take; and the results of these actions are then incorporated into one's self-model. If a self-model proves a generally bad guide to action selection, this may never be discovered, unless said self-model includes the knowledge that semi-random experimentation is often useful.

In what sense, then, may it be said that self is an attractor of iterated forward-backward synthesis? Backward synthesis infers the self from observations of system behavior. The system asks: What kind of system might we be, in order to give rise to these behaviors that we observe myself carrying out? Based on asking itself this question, it constructs a model of itself, i.e. it constructs a self. Then, this self guides the system's behavior: it builds new logical relationships its self-model and various other entities, in order to guide its future actions oriented toward achieving its goals. Based on the behaviors new induced via this constructive, forward-synthesis activity, the system may then engage in backward synthesis again and ask: What must we be now, in order to have carried out these new actions? And so on.

Our hypothesis is that after repeated iterations of this sort, in infancy, finally during early childhood a kind of self-reinforcing attractor occurs, and we have a self-model that is resilient and doesn't change dramatically when new instances of action – or explanation-generation occur. This is not strictly a mathematical attractor, though, because over a long period of time the self may well shift significantly. But, for a mature self, many hundreds of thousands or millions of forward-backward synthesis cycles may occur before the self-model is dramatically modified. For relatively long periods of time, small changes within the context of the existing self may suffice to allow the system to control itself intelligently.

This sort of system-theoretic speculation is difficult to validate scientifically, at the present stage of development of AI and cognitive science; yet, it is critical in terms of guiding the education of proto-AGI systems like the NCE as they interact with humans in environments like the NCE. The goal of educating the NCE in AGISim is not just to give it practical understanding of the world around it, but above all to give it a practical understanding of its own self, in relation to the world around it. By watching it do things, and explaining to it what it does; and having it watch others do things, and explain these things and react to them – in this way the baby AGI's systems self-model will originate and mature. The AGI's learning processes must be calibrated to allow both forward and backward synthesis operations to occur, specifically pertaining to patterns involving the system's perceptions of its own actions, behaviors and cognitions. In this way (assuming the cognitive operations are powerful enough and the environmental and social interactions are rich enough) the natural interaction of forward and backward cognition will lead to the emergence of an effective and growing self-model…

It is with this in mind that the Iterated Easter Egg Hunt scenario, described in Section 5 below, has been chosen for discussion and future experimentation. The goal has not been merely to choose a challenging learning problem (there are plenty of more challenging ones, of course), but rather to choose a learning problem that focuses specifically on the interaction between perception, cognition, action, and the pragmatic, contextual modeling of self and others. From this sort of interaction emerges the self; and from the actively evolving self-model emerge the seeds of more advanced autonomous general cognition.

4. An Economic Approach to Attention Allocation and Action Selection

This section briefly describes the economics-based approach to attention allocation and action selection contained in the NCE design. Along with PLN and MOSES, this is the third of the original and critical cognitive processes existing in the NCE. There are other cognitive processes, described above, that are critical but not that original in content: for instance, concept creation via blending, which is a very simple heuristic process of "cutting and pasting" Atoms; and map formation, which is essentially just conjunctive pattern mining applied to the Atom table. Although not yet in final form, PLN and MOSES have been fully implemented and experimented with, and are treated in other papers in this volume, as noted above. On the other hand, economic attention allocation and action selection have currently been implemented only in an incomplete prototype form. They will be discussed here only briefly and somewhat cursorily, the goal being to provide sufficient background that their role in the discussion of the iterated Easter Egg Hunt scenario in the following section may be understood. In terms of the above typology of cognitive processes, economic attention allocation underlies a number of processes: credit assignment, importance propagation, attention allocation, schema execution, and maintenance of the active schema and goal pools.

4.1. Economic Attention Allocation

This section outlines an approach to the allocation of processor time and memory to NCE Atoms based on the introduction of a concept of "currency" (money) into the NCE. The general advantages of currency-based attention allocation and credit assignment have been emphasized by Eric Baum in various papers (e.g. [23]; and see also discussion in [24]). The specific approach described here is not directly related to any of Baum's detailed ideas, but is grounded in the same philosophical ideas.

Two separate types of currency are introduced: STICurrency and LTICurrency, corresponding to short and long term importance values.[4] Atoms, MindAgents, and Units (the Unit, in the Novamente architecture, being a possibly distributed set of Novamente Lobes that are considered as sharing a common AtomTable for purposes of attention allocation) are then all considered as financial agents. The Units also have a unique role, as "mints" capable of producing new money. Money is not transferred between Units in this approach; each Unit has its own local, closed economy. It is assumed here, for sake of discussion, each Unit has a certain fixed total amount of currency of each type in it[5]. For starters, we may assume that this total amount is fixed for all eternity.

In the following discussion, the forgetting mechanism is assumed to be as follows: When the need to free up memory occurs, the Atoms with the least amount of LTI (i.e. of Currency of CurrencyType LTI) are removed from RAM. This is a simplistic forgetting mechanism which ignores subtler possibilities such as removing some links from some nodes with low LTI, but then letting them stay around a while in shrunken form.

On the other hand, the main point of STI is to govern the Atom-selection behavior of MindAgents. Many MindAgents choose Atoms to act upon based on their STI (STICurrency level). And, note that, in the economic approach described here, STI can pretty easily go to zero (or even become negative, representing "STI debt") if an Atom is useless for a while (i.e. an Atom can go "STI bankrupt"). If an Atom has negative

STI, it won't be selected by MindAgents using STI as a selection criterion, but it may receive STI from other Atoms related to it via the "importance spreading" dynamic, which may increase its STI to the level where it may once again be selected more frequently as an object of cognitive action.

Only the simplest variety of economic attention allocation is described here. More complexity is introduced when, for example, one introduces additional MindAgent-specific currencies, representing the STI of an Atom relative to the purposes of a given MindAgent. But these complexities are not needed for the discussion of goal and action selection and embodied social learning in the following sections, so they will be passed by for now.

4.1.1. Simple Equations for the Economics of Attention

The default equations for updating the amount of currency possessed by an Atom A, a MindAgent MA and a Unit U at a certain moment in time t are as follows.

First, equations for LTI currency:

```
   LTI_Atom(A,t+1) = LTI_Atom(A,t) - LTIAtomRent *
memoryUsage(A,t) + LTI_Atom_fee * (#times A used in cycle t)
+ LTI_Atom_rewards

   LTI_MindAgent(MA,t+1) = LTI_MindAgent(MA,t) -
LTI_fees_paid(MA,t) - processor_fees_paid(MA,t) +
rewards_received(MA,t)

   LTI_Unit(U,t+1) = LTI_Unit(U,t) +
LTI_rents_received(U,t) - LTI_rewards_paid(U,t)
```

Note that LTI values may go below zero, which is fine. In this case forgetting may remove the Atoms with the biggest debt; and if there are not enough Atoms in debt, it may remove some Atoms with positive LTI net worth as well.

Next, very similar but not quite identical equations for STI currency (the only difference is the lack of the memoryUsage term in the first equation):

```
   STI_Atom(A,t+1) = STI_Atom(A,t) - STIAtomRent +
STI_Atom_fee * (#times A used in cycle t) +
STI_Atom_rewards

   STI_MindAgent(MA,t+1) = STI_MindAgent(MA,t) -
STI_fees_paid(MA,t) -

   STI_Unit(U,t+1) = STI_Unit(U,t) +
STI_rents_received(U,t) - STI_rewards_paid(U,t)
```

Unlike with LTI, it seems best for Atoms with STI net worth <=T, where T is a specified threshold, to not be charged STIAtomRent. The argument is that if Atoms have such low net worth, they are not in the short-term memory in any useful sense, so they shouldn't have to pay for being in short-term memory. Letting Atoms accumulate a lot of STI debt would make the attentional focus sluggish to respond to new stimuli, destroying its ability to rapidly and spontaneously change focus. The threshold T is the "attentional focus boundary," whose existence induces an emergent "short term memory" within the overall AtomTable.

Note that in this approach, unlike in many other cognitive-science-based AI architectures, STM is not a separate system component but rather a systemic-dynamic phenomenon that emerges as an outgrowth of the dynamics of STI. This emergence is not magical but occurs because of specific choices in setting up the dynamics, i.e. the attentional focus boundary and its interaction with the rest of the economic attention allocation dynamics. But it has properties going beyond the existence of the attentional focus boundary: the setting of the boundary encourages the formation of complex strange attractors of attentional flow between entities that habitually surpass the attentional focus boundary at the same time.

Note also that the sets of equations for the LTI and STI currencies are totally separate from each other. This is intentional and represents a reasoned choice. In the currently proposed scheme, both LTI and STI currencies are proper, conserved currencies, but there is no mechanism for conversion between the two of them. What is not desired is for Atoms with high LTI to be able to purchase current attention just by virtue of having high LTI. Current attention must be purchased with the currency of recent utility (STI currency), whereas memory space must be purchased with the currency of long-term utility (LTI currency).

The meaning of the terms in the above equations will now be explained. I will first explain the equations for LTI currency, and then afterwards discuss STI currency (which is similar, but has the added complexity of Hebbian currency exchange).

4.1.2. LTI Economics

The LTIAtomRent is a an amount charged to each Atom each cycle, by the Unit, for the privilege of remaining in the AtomTable. This money is decremented from the Atom's currency balance and incremented to the Unit's currency balance. The LTIAtomRent is defined as the rent payable by an Atom per unit of memory usage, so that Atoms that are extremely consumptive of memory (for instance Atoms corresponding to large Combo trees in the ProcedureRepository) may be charged more total rent. In the initial version this dependency on memory usage may be omitted.

The LTIWage is the amount that a Unit pays a MindAgent for being utilized by a MindAgent. This enables an Atom to accumulate more currency if it is utilized more often by MindAgents. It seems optimal to enforce the rule that all Atoms must get paid the same wage. Note that, if Atoms could charge different wages (for example, based on their STI values), and wages were decremented from the MA's individual wealth stores directly, then MA's would be in the position of sometimes hiring inferior Atoms just to save money, and this would lead to suboptimal intelligence on the MA's part. It's true that this would serve to force competition between MA's, and that competition between MA's will be useful in future system versions where the system is evolving its own MA's. But even in these future systems, I think we can use better methods of enforcing competition among MA's, which do not involve artificially impairing the MA's intelligences.

Note that, in spite of charging the same fee, some Atoms will accumulate more LTI currency than others, because they will be selected by MA's more often than others. And the selection, by MA's, is based in large part on the ShortTermImportance (STI) quantity associated with Atoms. STI is not the only criterion for selection that MA's may use: in any particular instance, a MA may select Atoms based on any method it wants, which will often mean the use of criteria specific to the particular problem and context it has at hand. But as a default, once a MA has applied any other relevant selection filters, STI is the criterion it should use to select among the remaining Atoms available. This means that there is an influence relationship between STI currency and LTI currency, but it is an indirect relationship, not based on currency transfer. Of course, there is also an influence relationship in the reverse direction: LTI values affect STI values because if an Atom's LTI gets too low, it gets forgotten and therefore cannot get utilized and cannot accumulate any STI currency. So, just because there is no direct mechanism for transforming the two currency types into each other, doesn't mean the two are unrelated; it just means the relationship is not directly "financial."

Now let us look at economics from the MindAgent's perspective. When a MindAgent utilizes processor time, it must pay the Unit some of its currency in recompense for this time (and, symmetrically, the more time it uses, the more Atoms it will stimulate, therefore the more the Unit will pay out to Atoms on account of the MA's activity.) On the other hand, the MindAgents are paid by the Unit for contributing to system goals. This means that MindAgents that are generally more useful will be able to carry out operations involving more Atoms, and more processor time.

From the Unit's point of view, finally, revenue comes from the rents paid by Atoms, and funds are disbursed to Atoms in the form of rewards for utilization by MindAgents.

4.1.3. STI Economics

STI economics is basically the same as LTI economics, but with different parameter values. STI rents are higher, so that Atoms much more easily become STI-bankrupt; and the higher rents of necessity mean the other quantities in the economy must be different (Atoms must charge higher fees, so Units must give bigger rewards). And recall that STI rents are only charged to Atoms with positive STI net worth.

4.1.4. Hebbian Rewards

Next we introduce the notion of Hebbian rewards.

The basic idea is that Atoms pay other Atoms whose utilization is expected to pave the way for their own future utilization. That is, if there is a link

```
CausalHebbianLink A B
```

(denoting the fact that utilization of A seems to cause utilization of B) then it may be worthwhile for B to give some of its wealth to A – so as to (in the case of STI currency) increase the odds that A will be chosen by MA's, or (in the case of LTI currency) increase the odds that A will be retained in memory.

CausalHebbianLinks may come with time-interval stamps. STI Hebbian rewards should be given based on CHLinks with relatively brief time-interval stamps, whereas LTI Hebbian rewards should be given based on CHLinks with any time-interval stamp.

The total amount of wealth that an Atom should be willing to give to other Atoms at any point in time is capped (no sense to an Atom bankrupting itself to support others), and depends on the truth value of the CausalHebbianLinks that actually exist pointing to the Atom.

4.2. Economics of Goal and Action Selection

Now we will describe how these economic mechanisms are intended to interact with the processes of subgoal selection and action selection, in the NCE design. The main actors (apart from the usual ones like the AtomTable, economic attention allocation, etc.) in the tale to be told here are as follows:

- Structures:
  - Supergoal Pool
  - Active Schema Pool
- MindAgents:
  - GoalBasedSchemaSelection
  - GoalBasedSchemaLearning
  - GoalAttentionAllocation
  - FeasibilityUpdating
  - SchemaActivation

4.2.1. Supergoal Pool

The Supergoal Pool contains the Atoms that the system considers as top-level goals. These goals must be treated specially by attention allocation: they must be given funding by the Lobe so that they can use it to pay for getting themselves achieved. The weighting among different top-level goals is achieved via giving them differential amounts of currency. STICurrency is the key kind here, but of course top-level supergoals must also get some LTICurrency so they won't be forgotten. (Inadvertently deleting your top-level supergoals from memory is considered to be a bad thing!)

4.2.2. Promissory Transfer of STI Funds Between Goals

Transfer of "attentional funds" from goals to subgoals, and schema modules to other schema modules in the same schema, takea place via a mechanism of *promises of funding* (or "requests for service," to be called "RFS's" from here on). This mechanism relies upon and interacts with ordinary economic attention allocation but also has special properties.

The logic of these RFS's is as follows. If agent A issues a RFS of value x to agent B, then

1. When B judges it appropriate, B may redeem the note and ask A to transfer currency of value x to B.
2. A may withdraw the note from B at any time.

(There is also a little more complexity here, in that we will shortly introduce the notion of RFS's whose value is defined by a set of constraints. But this complexity does not contradict the two above points.) The total value of the of RFS's possessed by an Atom may be referred to as its "promise."

Now we explain how RFS's may be passed between goals. Given two predicates A and B, if A is being considered as a goal, then B may be considered as a subgoal of A (and A the supergoal of B) if there exists a relationship of the form

```
PredictiveImplication B A
```

I.e., achieving B may help to achieve A. Of course, the strength of this link and the temporal characteristics of this link are important in terms of quantifying how strongly and how usefully B is a subgoal of A.

Supergoals (not only top-level ones) allocate RFS's to subgoals as follows. Supergoal A may issue a RFS to subgoal B if it is judged that achievement (i.e., predicate satisfaction) of B implies achievement of A. This may proceed recursively: subgoals may allocate RFS's to subsubgoals according to the same justification.

Unlike actual currency, RFS's are not conserved. However, the actual payment of real currency upon redemption of RFS's obeys the conservation of real currency. This means that agents need to be responsible in issuing and withdrawing RFS's. In practice this may be ensured by having agents follow a couple simple rules in this regard.

1. If B and C are two alternatives for achieving A, and A has x units of currency, then A may promise both B and C x units of currency. Whomever asks for a redemption of the promise first, will get the money, and then the promise will be rescinded from the other one.
2. On the other hand, if the achievement of A requires both B and C to be achieved, then B and C may be granted RFS's that are defined by constraints. If A has x units of currency, then B and C receive an RFS tagged with the constraint (B+C<10). This means that in order to redeem the note, either one of B or C must confer with the other one, so that they can simultaneously request constraint-consistent amounts of money from A.

As an example of the role of constraints, suppose that the goal is to play fetch successfully (a subgoal of "get reward")… Then suppose it is learned that

```
ImplicationLink
   AND
         get_ball
         deliver_ball
   play_fetch
```

Then, if play_fetch has $10 in STICurrency, it may know it has $10 to spend on a combination of get_ball and deliver_ball. In this case both get_ball and deliver_ball would be given RFS's labeled with the contraint

```
RFS.get_ball + RFS.deliver_ball <= 10
```

The issuance of RFS's embodying constraints is different from (and generally carried out prior to) the evaluation of whether the constraints can be fulfilled.

A supergoal may rescind offers of reward for service at any time. And, generally, if a subgoal gets achieved and has not spent all the money it needed, the supergoal will not offer any more funding to the subgoal (until/unless it needs that subgoal achieved again).

As there are no ultimate sources of RFS in Novamente besides top-level supergoals, promise may be considered as a measure of "goal-related importance."

Transfer of RFS's among Atoms is carried out by the GoalAttentionAllocation MindAgent.

4.2.3. Feasibility Structures

Next, there is a numerical data structure associated with goal Atoms, which is called the feasibility structure. The feasibility structure of an Atom G indicates the feasibility of achieving G as a goal using various amounts of effort. It contains triples of the form (t, p, E) indicating the truth value t of achieving goal G to degree p using effort E. Feasibility structures must be updated periodically, via scanning the links coming into an Atom G; this may be done by a FeasibilityUpdating MindAgent. Feasibility may be calculated for any Atom G for which there are links of the form

```
Implication
   Execution S
   G
```

for some S. Once a schema has actually been executed on various inputs, its cost of execution on other inputs may be empirically estimated. But this is not the only case in which feasibility may be estimated. For example, if goal G inherits from goal G1,and most children of G1 are achievable with a certain feasibility, then probably G is achievable with that same feasibility as well. This allows feasibility estimation even in cases where no plan for achieving G yet exists, e.g. if the plan can be produced via predicate schematization, but such schematization has not yet been carried out.

Feasibility then connects with importance as follows. Important goals will get more STICurrency to spend, thus will be able to spawn more costly schemata. So, the GoalBasedSchemaSelection MindAgent, when choosing which schemata to push into the ActiveSchemaPool, will be able to choose more costly schemata corresponding to goals with more STICurrency to spend.

4.2.4. GoalBasedSchemaSelection

Next, the GoalBasedSchemaSelection selects schemata to be placed into the ActiveSchemaPool. It does this by choosing goals G, and then choosing schemata that are alleged to be useful for achieving these goals. It chooses goals via a fitness function that combines promise and feasibility. This involves solving an optimization problem: figuring out how to maximize the odds of getting a lot of goal-important stuff done within the available amount of (memory and space) effort. Potentially this optimization problem can get quite subtle, but initially some simple heuristics are satisfactory. (One subtlety involves handling dependencies between goals, as represented by constraint-bearing RFS's.).

Given a goal, the GBSS MindAgent chooses a schema to achieve that goal via the heuristic of selecting the one that maximizes a fitness function balancing the estimated effort required to achieve the goal via executing the schema, with the estimated probability that executing the schema will cause the goal to be achieved.

When searching for schemata to achieve G, and estimating their effort, one factor to be taken into account is the set of schemata already in the ActiveSchemaPool. Some schemata S may simultaneously achieve two goals; or two schemata achieving different goals may have significant overlap of modules. In this case G may be able to get achieved using very little or no effort (no additional effort, if there is already a schema S in the ActiveSchemaPool that is going to cause G to be achieved). But if G decides it can be achieved via a schema S already in the ActiveSchemaPool, then it should still notify the ActiveSchemaPool of this, so that G can be added to S's index (see below).

If the other goal G1 that placed S in the ActiveSchemaPool decides to withdraw S, then S may need to hit up G1 for money, in order to keep itself in the ActiveSchemaPool with enough funds to actually execute.

4.2.5. SchemaActivation

Next, what happens with schemata that are actually in the ActiveSchemaPool? Let us assume that each of these schema is a collection of modules, connected via ActivationLinks, which have semantics: (ActivationLink A B) means that if the schema that placed module A in the schema pool is to be completed, then after A is activated, B should be activated.

When a goal places a schema in the ActiveSchemaPool, it grants that schema an RFS equal in value to the (some fraction of) the (promissory+real) currency it has in its possession. The heuristics for determining how much currency to grant may become sophisticated; but initially we may just have a goal give a schema all its promissory currency; or in the case of a top-level supergoal, all its actual currency.

When a module within a schema actually executes, then it must redeem some of its promissory currency to turn it into actual currency, because executing costs money (paid to the Lobe). Once a schema is done executing, if it hasn't redeemed all its promissory currency, it gives the remainder back to the goal that placed it in the ActiveSchemaPool.

When a module finishes executing, it passes promissory currency to the other modules to which it points with ActivationLinks.

The network of modules in the ActiveSchemaPool is a digraph (whose links are ActivationLinks), because some modules may be shared within different overall schemata. Each module must be indexed via which schemata contain it, and each schema must be indexed via which goal(s) want it in the ActiveSchemaPool.

4.2.6. GoalBasedSchemaLearning

This, finally, refers to the process of trying to figure out how to achieve goals, i.e. trying to learn links between ExecutionLinks and goals G. This process should be focused on goals that have a high importance but for which feasible achievement-methodologies are not yet known. Predicate schematization is one way of achieving this; another is MOSES procedure evolution.

5. Embodied Social Learning in the Iterated Easter Egg Hunt Scenario

The goal of this section is to discuss how all the different aspects of the NCE design are intended cooperate to allow the system to carry out a moderately complex early-childhood-level task (iterated Easter Egg Hunt, or IEEH) in the AGISim world. The skeptical reader may be justified in viewing this section as a kind of highly technical science fiction, as the NCE has not yet been applied to this task, and will not be until a bit more development has been done. However, as argued above, I believe it is necessary to richly and deeply conceptualize the holistic behavior of an AGI system prior to designing its parts in detail (let alone implementing and testing its parts).

Note, it is not claimed that the approach described here is the optimal approach to solving the IEEH problem, either within the NCE or outside of it. Rather, IEEH is being used to exemplify the interaction of various cognitive mechanisms. An optimal

IEEH agent would likely learn much less from IEEH than either the NCE or a young human child.

An expected consequence of solving IEEH using a sophisticated cognitive approach (rather than, say, an operations-research or machine-learning approach) is superior generalization capability. For instance, suppose an NCE instance has learned how to play IEEH effectively using the general approach described here. Then it should have a much easier time learning to play hide-and-seek than an NM instance that has a similar background except that it has not learned how to play IEEH effectively. This kind of "transfer learning" is a key method of assessing the extent to which a task (like IEEH) has been learned in a way supporting general intelligence versus a narrow-AI/machine-learning sort of way (the latter tending to involve overfitting to the particular task). What one expects to see is that after a task T is learned, the learning of other tasks S becomes easier, with the degree of increased easiness being proportional to the similarity between S and T. While it is not clear what similarity measure is best used here, if we are aiming for roughly human-like intelligence then qualitative similarity according to human judgment is adequate.

5.1. Definition of Iterated Easter Egg Hunt

Firstly, "Easter Egg Hunt" is defined as a game in which one agent hides a bunch of eggs, and a group of other agents try to find them.

The main goal of each finder agent is to find as many eggs as possible. There may be other goals too, such as allowing each other agent to find at least one egg; or, finding more eggs than any other agent.

And, the main goal of the hider agent is to cause the finder agents to need to take a long time to find the eggs. Again, other goals may also be used in parallel, such as making it likely that each finder will get to find some eggs, rather than one seeker finding all the eggs.

Next, "Iterated Easter Egg Hunt" is defined as repetition of Easter Egg Hunt N times within a group of K agents, with different agents being the hider each time (the most interesting case is where N>K so everyone gets to be hider more than once).

5.2. Examples of Learning in IEEH

Useful patterns that may be recognized by an intelligent agent operating in an IEEH scenario include:

- Short agents cannot either hide or find eggs in very high places
- Short agents are more likely than tall ones to find eggs hidden in very low places (e.g. under a couch)
- Some agents may repetitively hide eggs in the same places each time they're serving as hider
- Some agents may try to hide eggs in different places each time they're serving as hider
- Once an agent A has found an egg in a certain place P, or seen another agent finding an egg in that place, then A is more likely to look in P again in the future

These patterns may be helpful to guide both hiding and finding behavior.

5.3. Pattern Mining, MOSES and Inference in IEEH

Many of the patterns discussed above may be found by pattern mining, and then validated by inference. In this subsection specific examples of this are described.

First of all, we may suppose that the system collects information such as

```
egg_134 found under couch_4 by agent_3
```

because it knows that "find" is a relevant predicate to the IEEH situation, so it tells the perception MindAgent to identify and record observed instances of "find"-ing.

If the system also has a general inclination to think about the effort levels being expended by agents, it may also collect information such as

```
egg_134 found under couch_4 by agent_3 with apparent
relative effort level LOW
```

(which might be expressed explicitly in Novamente Atoms, e.g., by

```
Evaluation [1]
   find
         agent_3
         egg_134
         SatisfyingSet
               Evaluation
                     under
                           $1
                           couch

Inheritance [2] Easter_Egg_Hunt
Inheritance [2] indoor

Context
   [2]
   Evaluation
         Effort
               [1]
               Low
)
```

Mining a collection of relationships of this form, using simple conjunctive pattern mining, may lead to patterns such as

```
Finding eggs hidden in drawers tends to be hard
```

The system may also record knowledge such as

```
agent_3 has height around 1 meter
```

as well, if it has the habit of recording physical properties of other agents.

Based on all this data, MOSES-driven pattern mining may easily discover the pattern

```
   Short agents have much higher odds of finding eggs hid-
den under the couch than tall agents do
```

It may also perhaps discover patterns such as

```
   Short agents have much higher odds of finding eggs hid-
den in the bottom drawer of a cabinet than tall agents do
```

(For this, MOSES would be given a fitness function defined by a measure of "statistical interestingness.")

Now, where does PLN come into the process? Well, the system may also have a concept of "low" (in terms of height) and then be able to learn from the above knowledge, via simple PLN inference, that

```
   Short agents have much higher odds of finding eggs hid-
den in low places than tall agents do
```

Next, supposing this statement has been learned via a combination of conjunctive pattern mining, MOSES and PLN, as described above. Then, it may be given a high importance value, because of its relevance to the current goals, and its surprisingly high truth value. Given this high importance, more attention will be focused on it.

Among other things that may happen because of this attention, inference will focus on the Atom, and resulting from this, generalization may occur. A good generalization would be

```
   Agents have relatively high odds of finding eggs that
they can relatively easily see
```

Further generalization may teach the system that

```
   RetroactiveImplication
      Evaluation find ($1, $2, *)
      Evaluation see ($1, $2)
```

(i.e., usually when an agent finds something, the agent has recently seen that thing.)

This may make the system assign the "see" predicate a high importance, which among other things may cause MOSES to focus on this predicate. The system has many examples of things it has seen and things it has not seen, and may mine this database of knowledge to learn patterns regarding seeing. It may note, for instance, that on several occasions it could not see a certain egg, and then when another agent moved some object that was in front of that egg, afterwards it could see that egg. If the system has a general category "agent" that abstracts both itself and the other agents, then it may learn from this the predicate

```
   If I move an object, I may see an egg that I did not
see before
```

This predicate may then, via the process of predicate schematization, be used to generate a schema that finds and moves objects, hoping to find eggs behind them.

5.4. Schema Execution in IEEH

Next, as an example of the role of multiple goals and schema execution in the Easter Egg Hunt scenario, let us consider the goals of:

- G1: Find as many eggs as possible
- G2: Find more eggs than anybody else

Suppose the NCE-controlled agent sees one egg (Egg_1) across the room, under the couch, with no one else evidently pursuing it; and sees two other eggs (Egg_2 and Egg_3) on a shelf across the room, and is not sure whether Agent_2 is pursuing Egg_2 or some other egg. Then

- G1 will spawn a schema S1 oriented toward retrieving Egg_2 and Egg_3
- G2 will spawn a schema S2 oriented toward retrieving Egg_1

Now there are a couple possibilities:

- Both schemata (S1 and S2) may be put into the ActiveSchemaPool at the same time, and given an amount of STICurrency commensurate with the importance of the parent goal.
- Predicate schematization may be asked to find a single schema serving the combined goal (G1 OR G2); and this single schema is then put in the ActiveSchemaPool

As an example of subgoaling, consider G2 above: find more eggs than anybody else. It may be difficult for the system to continually monitor how many eggs each other agent is finding. Thus, a good strategy would be for the system to learn implications such as

```
Implication (G3 AND G4) G2 <.9>
```

where

- G3 = find more eggs than Bill
- G4 = find more eggs than Bob

(which would be the case e.g. if Bill and Bob were generally the fastest egg-finders in the bunch.) Once this implication has been learned, the FeasibilityUpdating MindAgent has to notice it (which it will do, since it looks for potentially useful implications implying currently important goals), and then make feasibility evaluations regarding G3 and G4. A little inference will tell it that G3 and G4 are probably both more feasible (lower-cost) to achieve than G2, information that may then be recorded in the feasibility structures attached to these Atoms. Then, the GoalAttentionAllocation MindAgent will cause G2 to issue RFS's to G3 and G4; and the GoalBasedSchemaSelection MindAgent will quite likely select G3 and G4 to generate schemata to be placed in the ActiveSchemaPool.

5.5. Map Formation in the Understanding of Self and Others

Recall that one of the above inferences assumes the system has learned a notion of Agent that encompasses both itself and the other agents in the game. It's fair to assume that this abstraction has been learned prior to the system being able to grapple with a game as socially complex as IEEH. But still it's worth discussing how this learning may occur. This topic does not have to do with IEEH in particular, so it may be considered a kind of digression or appendix to the overall theme of IEEH.

Among other possible routes, this general notion of Agent may potentially be learned via pure unsupervised pattern mining.

First of all, the collection of body parts associated with a particular agent is a good example of a learned map: the body parts associated with some particular agent are all going to be associated with each other habitually, according to many different associations, and so a map should form for each one. (This is just an example of the role of map formation in "object recognition".) Mechanistically, these maps are initiall formed via the implicit activity of attention allocation and HebbianLink formation, which causes links to form between Atoms that are often utilized together in cognitive processes. Then the MapEncapsulation MindAgent will cause Atoms to form explicitly representing these maps, which means that the maps may be explicitly utilized within processes such as inference, pattern mining and MOSES.

So, suppose the agent has a body-map Atom for each of a number of other agents and also a body-map Atom for itself, then it has the opportunity to observe that all these body-maps are somewhat similar, and hence to cluster them together (via the Clustering MindAgent). Now there is a BodyMapCluster node, which may be studied analytically via inference and MOSES, leading to the emergence of explicit general knowledge regarding what constitutes a body, and the relations between aspects of bodies. For instance, it may be noted that when a body moves, this is generally associated with certain classes of leg and foot movements.

The system may then note that when its own body-map displays movement-associated leg and foot movements, this is associated with the execution of certain motoric schemata, internally. This merely requires conjunctive pattern mining, applied to a set of predicates involving both abstract predicates observed in the perceptual stream (the abstract predicate of movement-associated leg and foot movements), and also predicates involving motoric commands. What the system learns here is "When I carry out these particular actions, the result is that I carry out the action that I have identified as 'walking'." In other words, it has learned that certain leg and foot movements cause it to move in the same way that it observes other agents moving.

A host of different associations similar to this one may be mined, based on studying internal actions (mostly motoric, at this early stage) and their relationship to observed events involving the body-map. And, these various associations are often usefully considered together in reasoning about how to achieve various goals via coordinating actions. Because of these, these various associations will be Hebbianly associated – and ultimately the MapEncapsulation MindAgent may form a number of maps involving them. This is the root of the "self-model," as it exists in infantile embodied agents.

The use of this kind of self-model and other-model in the IEEH context should be fairly obvious. For instance, the system may observe that Agent_4 and Agent_7 are more similar to it body-map-wise than any of the other agents. If it has learned that agents with similar body-maps often carry out similar behaviors, then it may figure that it is better off trying to imitate what Agent_4 and Agent_7 do, egg-hunting-wise, rather than trying to imitate other agents. It may then focus its imitative efforts on imitating the best egg-finders, and also the egg-finders most similar to it.

6. Conclusion

Achieving artificial general intelligence at the human level and ultimately beyond is a large, ambitious task. Above, building on prior review papers, I have summarized some general aspects of the NCE design, and explained roughly how we intend them to work together in addressing a moderately complex early-childhood-level task, the iterated Easter Egg Hunt. / And I have professed the opinion that the cognitive processes required for this scenario are the same ones required for more complex, adult-level learning and reasoning.

More generally, why do I believe it is plausible to assert that the NCE design may actually be capable of achieving highly advanced general intelligence? The simplest answer consists of two very general points:

- The NCE is based on a *well-reasoned, truly comprehensive theory of cognition*, covering both the concretely-implemented and emergent aspects
- The specific algorithms and data structures chosen to implement this theory of mind are efficient, robust and scalable. And, so is the software implementation

A more nuanced answer refers to the system-theoretic ideas introduced in ([14,25,26]) and elaborated above: In the NCE design, forward and backward synthesis are implemented in a powerful and general enough way adequate to give rise to self and focused consciousness as strange attractors. This, is the crux of why, in my view, the NCE will very likely be able to give rise to powerful general intelligence. To yield powerful AGI, the "mechanics" of a system has to be right – procedures have to get executed, basic probabilistic conclusions have to be drawn, useless knowledge has to be forgotten, etc. But all this won't give rise to powerful intelligence unless the overall system is properly configured so as to give rise to the right emergent structures, key among which are the phenomenal self and the moving bubble of consciousness. NCE has been designed with this sort of emergence specifically in mind. Assuming the project continues as planned, the next years of work will tell us whether this methodology of emergence-oriented design, as instantiated in the NCE, is really as powerful as expected.

Endnotes

[2] A thorough technical review of the Novamente design has not yet been published; and whether and when such publication will occur is a subject of uncertainty. The main issue is not that the Novamente codebase is proprietary, but rather the issue of "AI safety" ([10,27–29]). Given the premise that the Novamente design may actually be capable of being used to create a human-level software intelligence, it then becomes ethically debatable whether the design should be placed in the public domain, due to the potential that someone may use the design to create a dangerous human-level software intelligence. Of course, going from a design to a working and educated implementation would be a large task for anyone, but the possibility is there, and has given us pause regarding the publication of too many details of the Novamente approach. On the other hand, we are still interested in sharing ideas with the research community and getting their feedback, and thus for the time being we have chosen the path of discussing the high-level aspects of the system design in papers such as this one, but not sharing technical details.

[4] An alternate approach involving only one currency was also considered. In this approach, STICurrency was taken as the basic currency, and the role of LTICurrency was played by the notion of a "credit rating." While interesting and workable, the added complexity of this approach was judged not worthwhile, in spite of the conceptual elegance of having only a single currency type.

[5] Later on, in a more advanced version of the NCE, this amount may eventually be allowed to change slowly over time – if it is found, for example, that periodically inserting more currency into the economy to

cause a small rate of inflation encourages intelligence. This would not be the case in the current system because none of the financial agents are initially carrying out economic actions with any real flexibility or intelligence (though their simple and mechanistic economic actions are enabling the system as a whole to carry out actions with some level of intelligence).

References

[1] Looks, Moshe (2006). Program Evolution for General Intelligence. Proceeding of AGI Workshop 2006, Bethesda MD, IOS Press.

[2] Goertzel, Ben, Ari Heljakka, Stephan Vladimir Bugaj, Cassio Pennachin, Moshe Looks (2006). *Exploring Android Developmental Psychology in a Simulation World.* Proceedings of ICCS-2006, Vancouver.

[3] Inhelder B. and J. Piaget. *The Growth of Logical Thinking from Childhood to Adolescence*. New York: Basic Books, 1958.

[4] Goertzel, Ben and Cassio Pennachin (2006). The Novamente Design for Artificial General Intelligence. In *Artificial General Intelligence*, Springer-Verlag.

[5] Goertzel, Ben (2006). *Patterns, Hypergraphs and General Intelligence*. Proceedings of International Joint Conference on Neural Networks, IJCNN 2006, Vancouver CA, to appear.

[6] Goertzel, Ben, C. Pennachin, A. Senna, T. Maia, G. Lamacie. (2003) "Novamente: an integrative architecture for Artificial General Intelligence." *Proceedings of IJCAI 2003 Workshop on Cognitive Modeling of Agents.* Acapulco, Mexico, 2003.

[7] Goertzel, Ben, C. Pennachin, A. Senna, M. Looks. (2004) "The Novamente Artificial General Intelligence Architecture." *Proceedings of AAAI Symposium on Achieving Intelligence Through Integrated Systems And Research.* Washington DC, 2004.

[8] Metzinger, Thomas (2004). *Being No One*. MIT Press.

[9] Goertzel, Izabela, Ben Goertzel, Ari Heljakka, Hugo Pinto and Cassio Pennachin (2006). Automated Biological Hypothesis Discovery via Probabilistic Inference on Dependency Grammar Parses of PubMed Abstracts, Proceeding of AGI Workshop 2006, Bethesda MD, IOS Press.

[10] Goertzel, Ben and Stephan Vladimir Bugaj (2006). Stages of Development in Uncertain-Logic-Based AI Systems, Proceeding of AGI Workshop 2006, Bethesda MD, IOS Press.

[11] Franklin, Stan (2006). A Foundational Architecture for Artificial General Intelligence. Proceeding of AGI Workshop 2006, Bethesda MD, IOS Press.

[12] Sloman, A. 1999. What Sort of Architecture is Required for a Human-like Agent? In *Foundations of Rational Agency*, ed. M. Wooldridge, and A.S. Rao. Dordrecht, Netherlands: Kluwer Academic Publishers.

[13] Ikle', Matthew, Ben Goertzel and Izabela Goertzel (2006). Indefinite Probabilities for General Intelligence, Proceeding of AGI Workshop 2006, Bethesda MD, IOS Press.

[14] Looks, Moshe (2006). Program Evolution for General Intelligence. Proceeding of AGI Workshop 2006, Bethesda MD, IOS Press.

[15] Goertzel, Ben (2006a). A System-Theoretic Analysis of Focused Cognition, and its Implications for the Emergence of Self and Attention. Dynamical Psychology.

[16] Fauconnier, Gilles and Turner, Mark (2002). *The Way We Think: Conceptual Blending and the Mind's Hidden Complexities.* Basic Books.

[17] Lenat, D. and R.V. Guha. (1990). *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project*. Addison-Wesley.

[18] Feldbaum, Christiane (1998) "WordNet: an electronic lexical database", Cambridge, The MIT press.

[19] Fillmore, Charles J., Collin F. Baker, and Hiroaki Sato. (2002). The framenet database and software tools. In Proceedings of the Third International Conference on Languag Resources and Evaluation, volume IV, Las Palmas. LREC.

[20] Goertzel, Ben (2006b). Lojban++: An Efficient, Minimally Ambiguous, User-Friendly Natural-Like Language for Human-Computer, Computer-Computer and Human-Human Communication, online at http://www.goertzel.org/papers/lojbanplusplus.pdf.

[21] Thelen, E. and L. Smith. (1994). *A Dynamic Systems Approach to the Development of Cognition and Action*. Cambridge, MA: MIT Press.

[22] Tomasello, Michael (2005). Constructing a Language. Harvard University Press.

[23] Baum, Eric (1998). "Manifesto for an Evolutionary Economics of Intelligence" in "Neural Networks and Machine Learning" Editor C. M. Bishop, Springer-Verlag (1998), pp. 285–344.

[24] Baum, Eric (2004). What Is Thought? MIT Press.

[25] Goertzel, Ben (1994). *Chaotic Logic*. Plenum, New York.

[26] Goertzel, Ben (1997). *From Complexity Creativity*. Plenum, New York.

[27] Bostrom, Nick (2002). Existential Risks: Analyzing Human Extinction Scenarios and Related Hazards. Journal of Evolution and Technology, vol. 9.
[28] Yudkowsky, Eliezer (2007). Cognitive Biases Potentially Affecting Judgment of Global Risks, in Global Catastrophic Risks, Ed. by Nick Bostrom and Milan Cirkovic, Oxford University Press.
[29] Yudkowsky, Eliezer (2007). Artificial Intelligence and Global Risk, in Global Catastrophic Risks, Ed. by Nick Bostrom and Milan Cirkovic, Oxford University Press.

Advances in Artificial General Intelligence: Concepts, Architectures and Algorithms
B. Goertzel and P. Wang (Eds.)
IOS Press, 2007
© 2007 The authors and IOS Press. All rights reserved.

Probabilistic Logic Based Reinforcement Learning of Simple Embodied Behaviors in a 3D Simulation World

Ari HELJAKKA[*], Ben GOERTZEL, Welter SILVA,
Cassio PENNACHIN, Andre' SENNA and Izabela GOERTZEL
Novamente LLC
[*]*heljakka@iki.fi*

Abstract. Logic-based AI is often thought of as being restricted to highly abstract domains such as theorem-proving and linguistic semantics. In the Novamente AGI architecture, however, probabilistic logic is used for a wider variety of purposes, including simple reinforcement learning of infantile behaviors, which are primarily concerned with perception and action rather than abstract cognition. This paper reports some simple experiments designed to validate the viability of this approach, via using the PLN probabilistic logic framework, implemented within the Novamente AGI architecture, to carry out reinforcement learning of simple embodied behaviors in a 3D simulation world (AGISim). The specific experiment focused upon involves teaching Novamente to play the game of "fetch" using reinforcement learning based on repeated partial rewards. Novamente is an integrative AGI architecture involving considerably more than just PLN; however, in this "fetch" experiment, the only cognitive process PLN is coupled with is simple perceptual pattern mining; other Novamente cognitive processes such as evolutionary learning and economic attention allocation are not utilized, so as to allow the study and demonstration of the power of PLN on its own.

1. Background and Motivation

The role of embodiment in AGI is somewhat philosophically controversial. Some claim that it is irrelevant: that, although human intelligence is closely tied to human embodiment, to extend this feature to AGI is unnecessary anthropomorphism. As alternatives to physical sensations and actions, suggested sources of knowledge for AGI include e.g. search engines like Google, and databases like Cyc [1]. Others, on the other hand, argue that embodiment is a necessary aspect of intelligence [2] – that mind is by nature embodied, and that disembodied software programs will never achieve true intelligence. Our view lies between these extremes. We view embodiment as an extremely convenient but not strictly necessary aspect of AGI systems.

One of the main advantages of embodiment for AGI is simply that humans are embodied, which implies that the study of embodied AGI's can benefit from our knowledge and intuition about embodied *human* intelligence. It is also clear that embodiment forces the deep integration of various aspects of intelligence – e.g. broad, shallow pattern recognition (needed for perception), procedure learning (needed for action), cognition, and attention allocation (needed for economical use of resources in a shifting environment). And, where learning human language is concerned, embodiment gives the opportunity for robust symbol grounding [3].

On the other hand, perhaps the biggest drawback of embodiment from the AGI developer's perspective is pragmatic in nature. Building, maintaining and using robots requires considerable effort and cost, and requires a different sort of expertise than AGI software development does. This lead to the proposal of using simulated embodiments existing in simulated 3D worlds. While dealing with simulation worlds also involves overhead in terms of time, cost and expertise, it is much milder in all these regards than physical robotics. No simulation world running on currently affordable hardware will be able to provide a fully accurate simulation of the perceptual and motor-control challenges faced by physical android robots. However, we suggest that contemporary simulation worlds, appropriately utilized, can nonetheless permit effective simulation of many of the cognitive challenges that physical robots face – and can provide a more-than-adequate environment for experimentation with embodied AGI.

With this philosophy in mind, we have created a 3D simulation world called AGISim [4], and begun using it to teach an AI system to control a simulated humanoid, in the context of interactions with a similar human-controlled humanoid. Within this framework we are pursuing an AI-teaching program loosely guided by Piagetan developmental psychology. Our current focus is on infant-level cognition, including basic phenomena such as the understanding of the permanence of objects and agents – and, the primary topic of the current paper, simple games like fetch, tag and hide-and-seek. The next phase of teaching will focus on "theory of mind" – on encouraging the AI system to come to its own understanding of the intentions and beliefs and knowledge of other cognitive agents, based on its interactions with them in the simulated world. Most of this paper will focus on a single example – how the Novamente AI system learns to play "fetch" in the AGISim simulation world. However, this example gains most of its interest and importance from its role as a small part of a larger picture involving embodied artificial cognitive development.

1.1. AGISim and Novamente

AGISim is being developed as an open-source project[1], led by the first two authors, and is based on the CrystalSpace[2] 3D game engine, which can be configured to display realistic physics. It allows AI systems and humans to control android agents, and to experience the simulated world via multiple senses, as well as having the capability to chat with each other directly through text. A similar approach with CrystalSpace was earlier used by John Santore and Stuart Shapiro, in using the "Sneps" paraconsistent logic system to control an agent called Crystal Cassie [5].

It is intended that the experience of an AGI controlling an agent in AGISim should display the main qualitative properties of a human controlling her body in the physical world. The simulated world should support the integration of perception, action and cognition in a unified learning loop. And, it should support the integration of information from a number of different senses, all reporting different aspects of a common world. With these goals in mind, we have created the initial version of AGISim as a basic 3D simulation of the interior of a building, with simulations of sight, sound, smell and taste. An agent in AGISim has a certain amount of energy, and can move around and pick up objects and build things. While not an exact simulation of any specific physical robot, the android agent an AI controls in AGISim is designed to bear suffi-

[1] sourceforge.net/projects/agisim.
[2] crystal.sourceforge.net.

cient resemblance to a simple humanoid robot to enable the fairly straightforward porting of control routines learned in AGISim to a physical robot.

Our work with AGISim to date has focused on controlling android agents in AGISim using the Novamente AI Engine (or NAIE; [6,7]), a comprehensive unique AI architecture that synthesizes perception, action, abstract cognition, linguistic capability, short and long term memory and other aspects of intelligence, in a manner inspired by complex systems science. Its design is based on a common mathematical foundation spanning all these aspects, which draws on probability theory and algorithmic information theory, among other areas. Unlike most contemporary AI projects, it is specifically oriented towards artificial *general* intelligence (AGI), rather than being restricted by design to one narrow domain or range of cognitive functions. The NAIE integrates aspects of prior AI projects and approaches, including symbolic, neural-network, evolutionary programming and reinforcement learning. The existing codebase is being applied in bioinformatics, NLP and other domains.

To save space, some of the discussion in this paper will assume a basic familiarity with NAIE structures such as Atoms, Nodes, Links, ImplicationLinks and so forth, all of which are described in previous references and in other papers in this volume.

1.2. Cognitive Development in Simulated Androids

Jean Piaget, in his classic studies of developmental psychology [8] conceived of child development as falling into four stages, each roughly identified with an age group: infantile, preoperational, concrete operational, and formal. While Piaget's approach is out-of-date in some ways, recent researchers have still found it useful for structuring work in computational developmental psychology [9]; we have modified the Piagetan approach somewhat for usage in our own work (see [10]). The basic Piagetan stages are as follows:

- Infantile: Imitation, repetition, association. Object permanence – infants learn that objects persist even when not being observed.
- Preoperational: Abstract mental representations. Word-object and image-object associations become systematic rather than occasional. Simple syntax.
- Concrete: Abstract logical thought applied to the physical world: conservation laws; more sophisticated classification; theory of mind – an understanding of the distinction between what I know and what others know. Classification becomes subtler.
- Formal: Abstract deductive reasoning contextually and pragmatically applied, the process of forming then testing hypotheses, etc.

We have carried out learning experiments involving the NAIE and AGISim, corresponding to aspects of Piaget's early stages of development.

We have obtained results which suggest that a Novamente-powered simulated android can learn, via its interactions with human-controlled simulated android agents, to carry out basic cognitive tasks like word-object association and understanding the permanence of objects and agents. In particular, for object permanence, we have created a simulation of the "A-not-B" task commonly used in developmental psychology (see e.g. [11]), in which Novamente's ability to solve this task is specifically tied to its correct use of a specific inference rule called the "Rule of Choice." The teacher hides an object in location A repeatedly, then eventually hides it in location B and asks the AI agent to find it. Human babies less than 9 months of age will often look in location A, but older babies look in B. The NAIE learns through interactive experience to look in location B – it learns that objects exist even when unobserved.

Another infantile example is the one that the bulk of this paper will focus on – “fetch”, the game commonly played by dogs and their masters. Dogs and small children, as well as Novamente, easily master this game. However, the use of probabilistic logic in a dynamic environment where the inference premises must be formed from perceptions in real-time requires some fairly sophisticated probabilistic inference, as well as complex integration of other AI processes running in parallel. There are many other ways Novamente could learn to perform this task, e.g. using MOSES, or combining PLN and MOSES in various ways. It is not surprising that, given a simple task like fetch and a complex architecture like Novamente, there should be an oversupply of cognitive methods for solving the problem.

2. The Fetch Task

The basic idea underlying “fetch” is a simple one: the human throws an object and says “fetch,” the dog runs to the object, picks it up, and brings it back to the human, who then rewards the dog for correct behavior. In our learning experiments, the teacher (a humanoid agent in AGISIm) plays the role of the human and the Novamente-controlled agent plays the role of the dog.

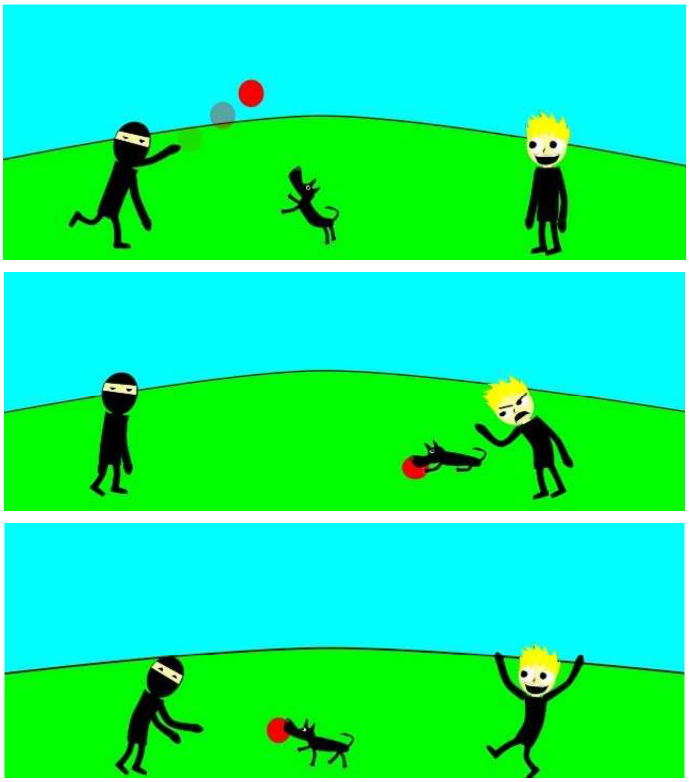
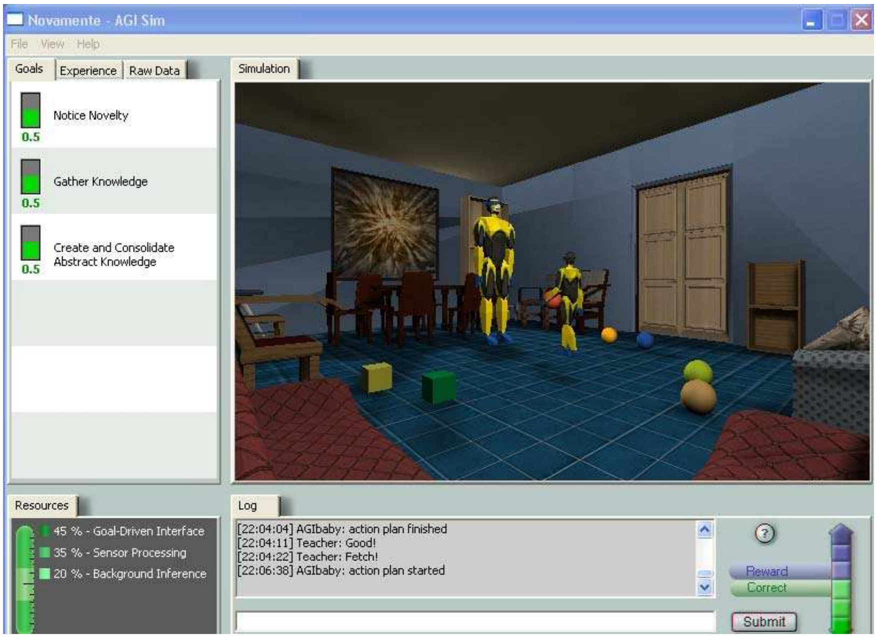


Figure 1. Dogs playing fetch, correctly (bottom) and incorrectly (middle). (Illustrations by Zebulon Goertzel).



```
[22:04:04] AGIbaby: action plan finished
[22:04:11] Teacher: Good!
[22:04:22] Teacher: Fetch!
[22:06:38] AGIbaby: action plan started
```

Figure 2. A screenshot of Novamente and its teacher playing fetch in the AGISim simulation world. Here Novamente is returning the ball to the teacher, after the teacher has thrown it.

In more complex AGISim experiments, the teacher is actually controlled by a human being, who delivers rewards to Novamente by using the Reward controls on the AGISim user interface. In the fetch experiments reported here, because the task is so simple, the human controller was replaced by automated control code. The critical aspect of this automated teacher is partial reward. That is, you can't teach a dog (or a baby Novamente) to play fetch simply by rewarding it when it successfully retrieves the object and brings it to you, and rewarding it not at all otherwise – because the odds of it ever carrying out this "correct" behavior by random experimentation in the first place would be very low. What is needed for instruction to be successful is for there to be a structure of partial rewards in place.

We used here a modest approach with only one partial reward preceding the final reward for the target behaviour. The partial reward is given, first, whenever the agent manages to lift the thrown object. This reward is repeated several times, in order to make the agent learn that this behaviour implies reward, causing the agent to repeat this behaviour for some time even once the reward is removed, followed by some random actions. Finally, every time the rewarded behaviour happens to be followed by the agent carrying the object to the teacher and dropping it on her feet, the agent is rewarded again. In this manner, the agent learns the sequence of actions required for playing the game of "fetch." Given a more complex system of partial rewards, the agent can likewise be made to learn similar behaviour in a richer environment and with a larger set of possible actions.

3. Learning Fetch Within the Novamente Architecture

Novamente is an integrative AGI architecture, in which the highest levels of intelligence are intended to be achieved via the combination of several cognitive processes. However, fetch is a very simple task and does not require the full force of even the current incomplete version of the integrated Novamente cognition. From a Novamente point of view, it is interesting mostly as a "smoke test" for embodied reinforcement learning, to indicate that the basic mechanisms required for cognitive interaction with AGISim are integrated adequately and working correctly.

As noted above, fetch can be learned in Novamente in more than one way. The basic learning can be driven for example by PLN probabilistic reasoning or by MOSES evolutionary learning or by various combinations of aspects of the two. Here we will discuss only the PLN approach in depth.

3.1. Some Comments on Object Recognition

Firstly, in either the PLN or MOSES cases, learning fetch requires the assumption that the system already knows how to recognize objects: balls, teachers, and the like. I.e., the system must have learned (or been provided intrinsically with the capability for) "object recognition." We will make only a few comments about this here, as it is not the focus of this paper. Basically, object recognition, as the term is intended here, is the process of taking in raw data from the perceived environment (the AGISim world), and identifying in it collections of percepts that constitute "persistent objects." For instance, if the agent is looking at a scene consisting of a ball and a box, it needs to be able to identify that it is in fact looking at a ball and a box. And, if the ball rolls to the right slightly, it need to be able to identify that in some sense it is still looking at the same ball and the same box. This sort of capability develops fairly early in humans, but appears not to fully exist in very young babies. It seems very likely that its development in humans is a combination of learning with the progressive unfolding of genetically-programmed capabilities. Object recognition is not intrinsically dependent upon vision (see e.g. [12] for a treatment of the development of object recognition and related capabilities in blind babies), but in sighted people it is mainly dependent on vision, and the way we have handled it in AGISim so far is heavily vision-centric.

AGISim supports varying levels of visual granularity[3], including

- voxel vision: the system sees specific colors existing at specific coordinate points in the sim world.
- polygon vision: the system sees specific polygons with specific colors, and with corners at specific coordinate points in the sim world.
- object vision: the system sees specific polyhedral objects with corners at specific coordinate points.

[3] From a robotics vision perspective, it must be noted that even AGISim voxel vision is "cheating," in the sense that it involves the AI system directly receiving information about the distances of perceived voxels from its eyes. The human brain must infer distances using stereo vision, a complex matter in itself. However, some robots currently infer distances using lidar rather than stereopsis. In any case, in the Novamente project we have chosen not to get involved with low-level vision processing issues of this nature. When the time comes to interface Novamente with a physical robot, our approach will likely be to integrate an external vision-processing module that supplies either voxel or polygon vision inputs of the sort described above.

The object-recognition problem is restricted to the first two cases. We have not dealt with the voxel vision case so far, but have decided that polygon vision represents a nice intermediary level: low enough to present the system with basic problems such as object recognition, yet high enough not to lead to the full computational complexity of human-style vision processing. Object recognition in the polygon vision case is a relatively simple matter of conjunctive pattern mining and clustering, but the specific algorithms associated with this process in Novamente will not be discussed here as they would lead us too far afield.

3.2. Inference, Pattern Mining and Predicate Schematization

Given object recognition, MOSES can learn to play fetch right away, without the need for any auxiliary learning processes. PLN could in principle do the same, with the help of an additional cognitive mechanism we call “predicate schematization.” This process converts the logical knowledge obtained by PLN regarding how to play fetch into concrete procedural knowledge that can be executed. Planning is a familiar manifestation of the general mechanism.

However, in order to make PLN learning of the fetch behavior reasonably efficient, an additional cognitive mechanism called “pattern mining” is also needed. This is a process that identifies frequent or otherwise significant patterns in a large body of data. The process is independent of whether the data is perceptual or related to actions, cognition etc. In principle, everything obtained via pattern mining could also be obtained via inference, but pattern mining has superior performance in many applications.

In either the PLN or MOSES approach, finally, the action of the Novamente component that actually executes learned procedures (called *schemata* in Novamente) is required.

So, in sum, the two minimal approaches to learning fetch in Novamente may be described as follows:

- object recognition + MOSES + schema execution
- object recognition + pattern mining + PLN + predicate schematization + schema execution

Neither of these approaches is optimal; for instance, greater learning efficiency may be obtained by integrating prioritization heuristics, which in Novamenre design are known as *economic attention allocation*, with MOSES and/or PLN so that the system wastes less time focusing on irrelevant things. But given the simplicity of the task, either of these approaches is sufficient. For this particular task, MOSES is a simpler but computationally slower approach than PLN; but that is really irrelevant, as our point here is not focused on the fetch task in itself (which obviously could be solved via much simpler methods than anything in Novamente) but on what it teaches us about the use of probabilistic inference in the context of embodied reinforcement learning.

4. Pattern Mining in Novamente

The “pattern mining” step mentioned above has not been discussed in previous publications on Novamente. We mention it briefly here. The pattern mining component is at the moment very simple, but may be made more sophisticated in future. The inputs of pattern mining are, in general,

- Atoms denoting raw outputs of the “sensors” that Novamente possesses in the AGISim world
- Atoms indicating actions Novamente has taken, e.g. in the AGISim world
- Potentially, other Atoms in Novamente’s memory

The basic principle underlying pattern mining is to find *noteworthy combinations of inputs*. Furthermore, the mining process must find these via a purely “greedy search” methodology, without any slow and exploratory searching such as occurs for instance in evolutionary learning mechanisms like MOSES. Pattern mining has to operate in real time on potentially very large volumes of data (although in the current AGISim configuration, the amount of data is in fact not very large due to the relatively small number and simpliciy of objects in the environment).

The current Novamente pattern miner operates via a simplistic “conjunction mining” approach. It is supplied with a set of standard *perceptual predicates*, which may be applied to its inputs. It then searches for conjunctions of these perceptual predicates, which occur surprisingly often in its experience.

For the fetch learning example, the perception process is relatively simple, and focused on the recognition of temporal patterns, such as those to be discussed in the following subsection.

4.1. Mining Temporal Patterns Relevant to Playing Fetch

All of the pattern mining relevant to the fetch learning example is temporal in nature. Frequent sequences of events must be recognized. As a simple example, sequences such as

```
SequentialAND
   SimultaneousAND
         I am holding the ball
         I am near the teacher
   I get more reward
```

Must be recognized. More formally, this looks like

```
SequentialAND
   SimultaneousAND
         holding(ball)
         near(me, teacher)
   Reward
```

Or in full-fledged Novamente Node/Link notation,

```
SequentialAND
   SimultaneousAND
         EvaluationLink holding ball
         EvaluationLink
               near
               ListLink (me, teacher)
   Reward
```

The predicates required here are near() and holding(), as well as the primitive "sensation" of Reward. Given these predicates as primitives, the mining of this conjunction from the system's experience is a simple matter. A similar approach works for mining conjunctions regarding other phases of the partial reward schedule used for fetch, as will be detailed below.

4.2. Intended Improvements to Novamente's Perceptual Pattern Mining Architecture

As a minor digression, it is anticipated that the simplistic approach to perceptual pattern mining discussed above may not hold up when we begin experimenting with a richer environment within AGISim (i.e. with environments containing a large number of complex objects); and so there is also a more complex design, not yet implemented, that involves embedding the conjunctive miner within a hierarchical architecture. Essentially, one may construct a hierarchical perception network (a subnetwork of Novamente's overall Atom network) in which each node refers to a certain localized region of spacetime, with the children of a node corresponding to the subregions of the region the node corresponds to. One may then carry out conjunctive mining within each local node of the hierarchical perception network.

Philosophically, this hierarchical perception approach displays significant similarities to Jeff Hawkins' [13] hierarchical perception architecture, though without his attempts at neurological justification. Hawkins' architecture relies on a combination of neural net activation spreading and Bayesian network based probabilistic calculations. On the other hand, in Novamente, we may spread attention between nodes in the perception network using economic attention allocation (similar to Hawkins' neural net activation spreading); and we may update the probabilities of conjunctions at various levels in the hierarchy using PLN probabilistic inference, which is ultimately just a different rearrangement of the mathematics used in Bayes nets (though PLN's arrangement of probability theory has significantly more general applicability). It may be noted that this combination of attentional and probabilistic dynamics also characterizes Maes behavior nets [14], which have some parallels to the action selection mechanisms in both Novamente and Stan Franklin's LIDA system [15]. It is with this kind of combination in mind that all Novamente Atoms have been supplied with both truth values and attention values.

5. PLN in the Fetch Example

As a prior chapter in this book [16] has already given a basic overview of PLN inference, this material will not be repeated here. Rather, a high-level overview of the application of PLN to the fetch problem will be given, together with a brief discussion of two relevant aspects of PLN that were not covered in the other chapter: inference about actions, and temporal inference.

From a PLN perspective, learning to play fetch is a simple instance of backward chaining inference. The goal of Novamente in this context is to maximize reward, and the goal of the PLN backward chainer is to find some way to prove that if some actionable predicates become true, then

```
Evaluation (Reward)
```

becomes true. This inference is possible by assuming that trying out actions is always possible, ie. the actions are considered to be in the axiom set of the inference. A more elegant approach we did not try yet would be to set a

```
PredictiveImplicationLink($1, Reward)
```

as the target of the inference, and launch the inference to fill in the variable slot $1 with a sequence of actions.

PredictiveImplicationLink is a Novamente Link type that combines logical (probabilistic) implication with temporal precedence. Basically, the backward chainer is being asked to construct an Atom that implies the future obtaining of reward. Each PredictiveImplicationLink contains a time-distribution indicating how long the target is supposed to occur after the source does; in this case the time-distribution must be centered around the rough length of time that a single episode of the "fetch" game occupies.

To learn how to play fetch, Novamente must repeatedly invoke PLN backward chaining on a knowledge base consisting of Atoms that are constantly being acted upon by perceptual pattern mining as discussed above. PLN learns logical knowledge regarding what circumstances imply reward, and then the predicate schematization process produces executable schemata embodying this knowledge, which are then executed – causing the system to carry out actions, which lead to new perceptions, which give PLN more information to guide its reasoning and lead to the construction of new procedures, etc.

The representation of temporal knowledge used in PLN (and Novamente generally) is based on a variant of the Event Calculus approach [17]. Here, we use a restricted set of predicates, only including the PredictiveImplicationLink and SequentialAnd(A B), which has the semantics "the event that A occurs, and then after A terminates, B initiates". For instance in the case of

```
PredictiveImplicationLink A B
```

the truth value denotes (roughly speaking) the probability

```
P(event in class B initiates |
               event in class A terminates previously)
```

or more explicitly the average of w(x,y) over all (x,y) so that x is an event in A and y is an event in B, where w(x,y) is a "time distribution function" so that

- w(x,y)= 0 if x initiates after y terminates
- w(x,y) =1 if y's initiation is simultaneous with x's termination
- w(x,y) depends monotonically on the difference diff=(initiation of y – termination of x)

For example, in many cases one may use

```
w(x,y) = k/(diff+k)
```

where k is an adjustable parameter. For more details on temporal links and the event calculus variant we use, see [18].

In order to carry out very simple inferences about schema executions as required in the fetch example, two primitive predicates are used by PLN in Novamente:

- *try*, where try(X) indicating that the schema X is executed
- *can*, where can(X) indicates that the necessary preconditions of schema X are fulfilled, so that the execution of X will be possible

Furthermore, the following piece of knowledge is assumed to be known by the system, and is provided to Novamente as an axiom:

```
PredictiveImplication
   SimultaneousAnd
         Evaluation try X
         Evaluation can X
   Evaluation done X
```

That is: if the system can do X, and it tries to do X, then it has done X. Note that this implication may be used probabilistically, so it can be applied e.g. in cases where it is not certain whether or not the system can do X or not. If the system is unsure whether it can do X, then it will (according to this implication, without any additional knowledge) be unsure as to whether X will be done even if it tries to do X; and PLN's uncertain inference formulas may be used to estimate the latter uncertainty.

The proper use of the "can" predicate necessitates that we mine the history of occasions in which a certain action succeeded and occasions in which it did not. This allows us to create PredictiveImplications that embody the knowledge of the preconditions for successfully carrying out an action. In this paper we use a simpler approach because the basic mining problem is so easy: we just assume that "can" holds for all actions, and push the statistics of success/failure into the truth values of the PredictiveImplications produced by pattern mining. Hence, we don't try to determine the situation which enables the agent to carry out an action, but simply observe that in a certain percentage of cases the action succeeded, specifically in cases where it was part of a longer sequence of actions, so that the ealier actions likely contributed or fulfilled to the preconditions of the later action. We will soon illustrate this in the context of a real inference trail.

5.1. Inference Rules Used For Learning to Play Fetch

Finally, this section enumerates and explains the inference rules used in learning to play fetch.

Firstly, the ModusPonensRule is unsurprisingly a probabilistic version of modus ponens, i.e.

```
Implication A B
A
|-
B
```

Modus Ponens can also be applied to PredictiveImplications, insofar as the system keeps track of the structure of the proof tree so as to maintain the proper order of arguments. It is perhaps fortuitous that the order in which the arguments to Modus Ponens must be applied is always the same as the related temporal order, which allows us to extract a plan of consequtive actions, in an unambiguous order, from the proof tree.

Relatedly, the AndRule is of the form

```
A
B
|-
A & B
```

The AndRule can be accompanied with a temporal truth value formula so as to make them applicable for creating SequentialANDs. Later, we will only be concerned with SimpleANDRule, which is the AndRule that only looks at its constituents individually, and does not try to take advantage of partial conjunctions that could hold useful information for the process of estimating the truth value of the complete conjunction.

The DeductionRule is of the form

```
Implication A B
Implication B C
|-
Implication A C
```

The PLN truth value formulas for these three rules are given in [18], in this volume.

The RewritingRule is a composition of AndRule and ModusPonensRule. It is used as a shorthand for converting atoms from one form to another when we have a Boolean true implication at our disposal. For these implications, the use of probabilistic inference is obviously unnecessary.

Finally, the function of the CrispUnificationRule is simply to produce, from a variable-laden universally quantified expression (atom), a version in which one or more variables has been bound. The truth value of the resulting atom is the same as that of the quantified expression itself.

6. Learning to Play Fetch via PLN Backward Chaining

This section describes one specific logical plan learned by PLN, based on the output of the perception miner, in order to play fetch in the AGISim world. The Novamente system is nondeterministic and different runs of the system, given the same environment and reward scheme, may lead to a variety of different internal plans and schemata. The learned plan discussed here is a representative one that lends itself relatively well to discussion.

Firstly, we define the specific predicates used as primitives for this learning experiment:

- Reward – a built-in sensation corresponding to the Novamente agent getting Reward, either via the AGISim teaching interface or otherwise via having its internal Reward indicator stimulated
- goto – a persistent event: goto(x) means the agent is going to x
- lift – an action: lift(x) means the agent lifts x
- drop – an action: drop(x) means the agent lifts x (and when this happens close to an agent T, we can interpret that informally as "giving" x to T)
- TeacherSay – a percept, TeacherSay(x) means that the teacher utters the string x
- holding – a persistent event, holding(x) means the agent is holding x

By assuming the above predicates, we are abstracting away from any actual motor learning: we are assuming that the system already knows how to lift and hold and drop, for example. Novamente's learning mechanisms appear to be capable of learning procedures to ground these motor actions, particularly in the simple context of the AGISim simulated robot, but these motor-learning experiments have not yet been done, so for the purpose of the fetch experiments reported here, we have simply assumed hard-coded procedures for these motor actions.

Also, note that we have assumed goto(x) as a built-in behavior. This is psychologically realistic, in the sense that before a dog or baby learns to play fetch, they have certainly already learned to move to an object that interests them. Learning goto(x) in the context of a very simple environment like the one used for these fetch experiments is very easy for Novamente, and unlike the motor learning experiments, this learning experiment has already been done. So the assumption made here is basically just that the fetch experiment must be done in a Novamente system that has already learned goto via prior teaching or spontaneous learning.

6.1. A Multi-Stage Partial Reward Function for Learning to Play Fetch

Next, we describe the partial reward function as a set of logical implications. The partial reward function is broken down into two stages, as described above. In these and following examples we use a shorthand notation, designed to improve legibility, e.g. PredImp for PredictiveImplicationLink, and "done goto ball" instead of

```
EvaluationLink
   done
   EvaluationLink goto ball
```

etc. The two stages of the reward function are:

Stage 1
```
PredImp
   holding ball
   Reward
```

Stage 2

```
PredImp
   SeqAnd
         holding ball
         done goto teacher
         done drop ball
   Reward
```

6.2. Knowledge Gained via Pattern Mining

Next, in the course of attempting to get rewarded by fulfilling the above partial reward functions, the pattern mining subsystem recognizes a number of conjunctions which PLN then evaluates and turns into implications. An example of the resulting implications is the following:

```
PredImp
   SeqAnd
         done goto Ball
         done lift Ball
   holding Ball
```

6.3. A PLN Inference Trajectory for Learning to Play Fetch

Next, this bulk of this section shows a PLN inference trajectory which results in learning to play fetch according to the Stage 4 reward function mentioned above. This trajectory is one of many produced by PLN in various learning runs. When acted upon by the predicate schematization process, it produces the simple schema (executable procedure)

```
try goto Ball
try lift Ball
try goto Teacher
try drop Ball
```

It is quite striking to see how much work PLN and perception need to go through to get to this relatively simple plan! In fact, MOSES evolutionary learning can find it more simply, because the plan itself is quite small. However, MOSES occupies more runtime searching for the plan than PLN does, because MOSES essentially finds this plan through guided random search, rather than through understanding of the problem.

In a sense, MOSES has an easier time of it – MOSES doesn't have to worry about the nature of "holding" at all, or the distinction between ongoing events and actions. However, in the process of learning this program, MOSES also does not build up as much knowledge that is useful for solving other problems. PLN and perceptual pattern mining learn to play fetch by understanding each part of the "fetch" game and why it helps get partial reward, and then piecing together the behaviors corresponding to the different parts of the game into an overall plan. In the context of learning more complex tasks, of course MOSES and PLN may work together providing superior intelligence to what may be produced by either one separately. But in the context of this simple task of fetch, PLN can do the learning job quite efficiently with support only from pattern mining, without needing support from more sophisticated pattern recognition tools like MOSES.

The final inference trajectory follows.

First of all, the inference target was:

```
EvaluationLink (77) <0.80, 0.0099>  [4069]
   Reward:PredicateNode (26) <1, 0>  [191]
```

Note the truth value of the EvaluationLink initially found, which is <0.80, 0.0099>. Referring back to the definition of PLN truth values, this means that the inference process, after it had been running for suffiently long, found a way to achieve the Reward with a strength of 0.80, but with a weight of evidence of only .0099 (the rather unforgiving scaling factor of which originates from the internals of the perception miner). Continuing the run makes the strength increase towards, but not achieve, 1.0.

This target was produced by applying ModusPonensRule to the combination of

```
PredictiveImplicationLink <0.8,0.01> [9053948]
   SequentialAndLink <1,0> [9053937]
      EvaluationLink <1,0> [905394208]
         "holdingObject":PredicateNode <0,0> [6560272]
         "Ball":ConceptNode <0,0> [6582640]
      EvaluationLink <1,0> [905389520]
         "done":PredicateNode <0,0> [6606960]
         ExecutionLink [6888032]
               "goto":GroundedSchemaNode <0,0> [6553792]
               "Teacher":ConceptNode <0,0> [6554000]
      EvaluationLink <1,0> [905393440]
         "try":PredicateNode <0,0> [6552272]
         ExecutionLink [7505792]
               "drop":GroundedSchemaNode <0,0> [6564640]
               "Ball":ConceptNode <0,0> [6559856]
   EvaluationLink <1,0> [905391056]
      "Reward":PredicateNode <1,0> [191]
```

and

```
SequentialAndLink <1,0.01> [840895904]
   EvaluationLink <1,0.01> [104300720]
      "holdingObject":PredicateNode <0,0> [6560272]
      "Ball":ConceptNode <0,0> [6582640]
   EvaluationLink <1,1> [72895584]
      "done":PredicateNode <0,0> [6606960]
      ExecutionLink [6888032]
            "goto":GroundedSchemaNode <0,0> [6553792]
            "Teacher":ConceptNode <0,0> [6554000]
   EvaluationLink <1,1> [104537344]
      "try":PredicateNode <0,0> [6552272]
      ExecutionLink [7505792]
            "drop":GroundedSchemaNode <0,0> [6564640]
            "Ball":ConceptNode <0,0> [6559856]
```

Graphically, the previous two link constructs would be denoted

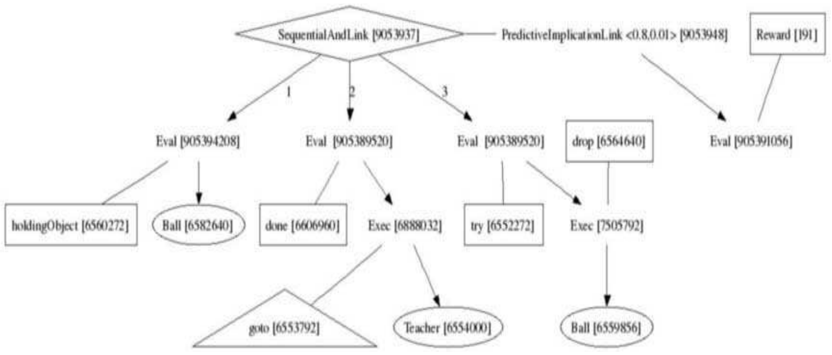


Figure 3.

and

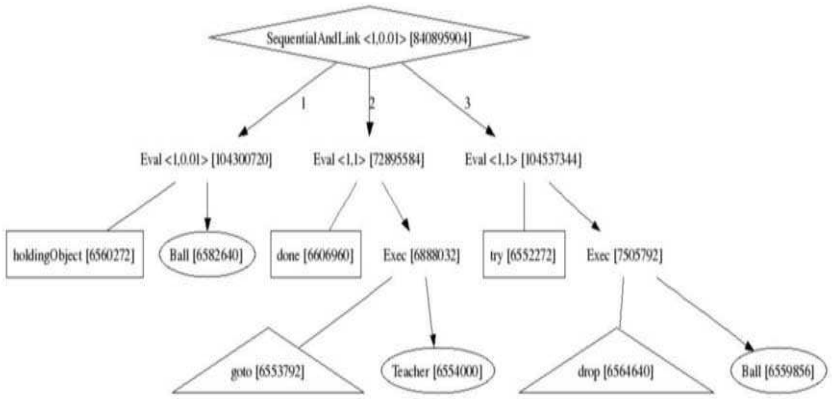


Figure 4.

Respectively. In the rest of this discussion we will often substitute graphical depictions for the indent-notation, in the interest of increasing comprehensibility.

Next, the SequentialANDLink [840895904] was produced by applying SimpleANDRule to its three child EvaluationLinks.

The EvaluationLink [104300720] was produced by applying ModusPonensRule to:

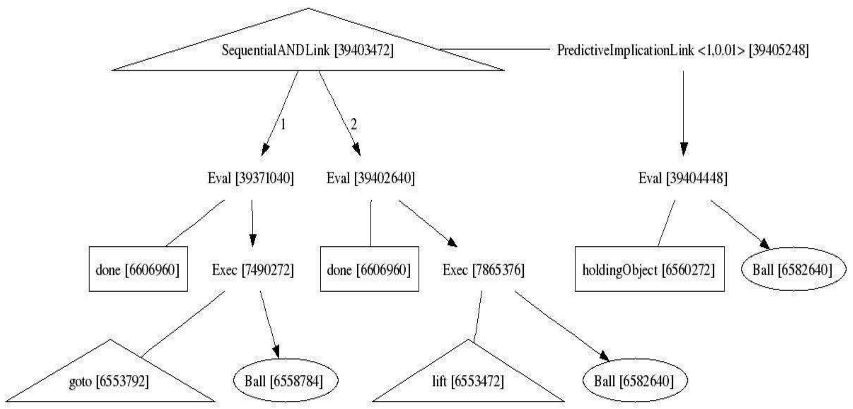


Figure 5.

which was mined from perception data, and to

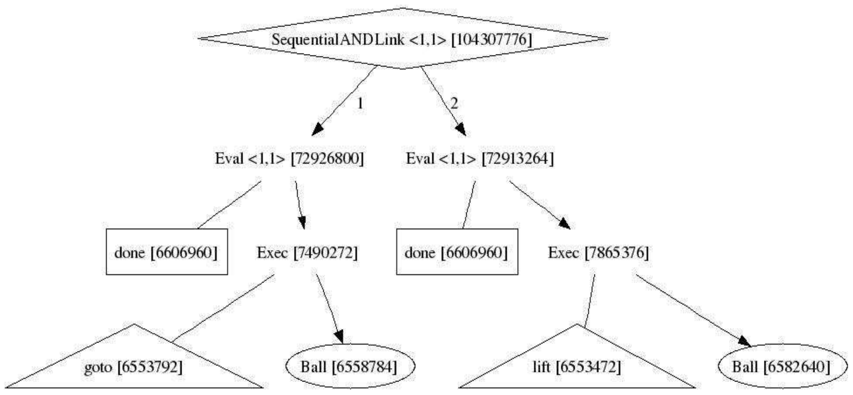


Figure 6.

The SequentialANDLink [104307776] was produced by applying SimpleANDRule to its two child EvaluationLinks. The EvaluationLink [72926800] was produced by applying RewritingRule to:

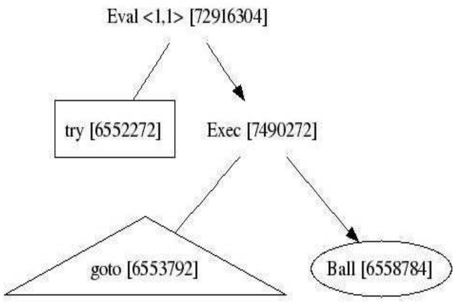


Figure 7.

and

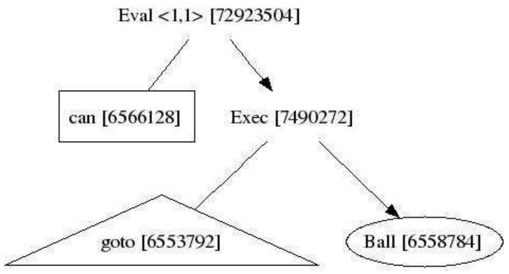


Figure 8.

The EvaluationLink [72916304], as well as all other *try* statements, were considered axiomatic, and technically produced by applying CrispUnificationRule to:

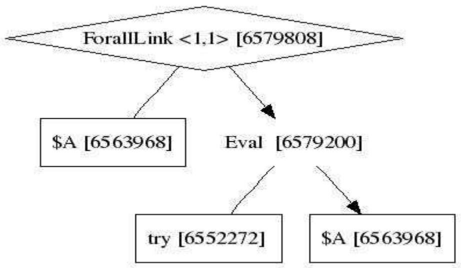


Figure 9.

The EvaluationLink [72923504], as well as all other *can* statements, were considered axiomatic, and technically produced by applying CrispUnificationRule to:

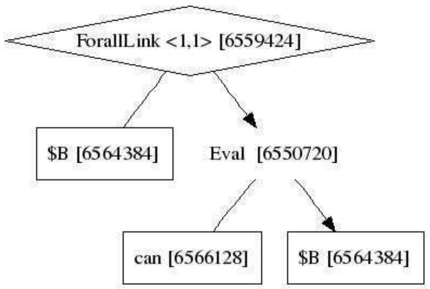


Figure 10.

The EvaluationLink [72913264] was produced by applying RewritingRule to:

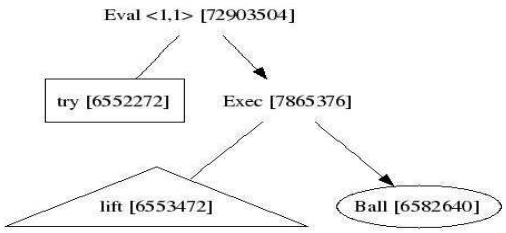


Figure 11.

and

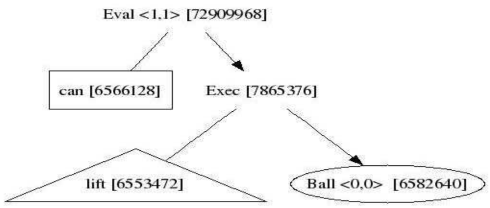


Figure 12.

Returning to the first PredictiveImplicationLink's children, EvaluationLink [72895584] was produced by applying RewritingRule to:

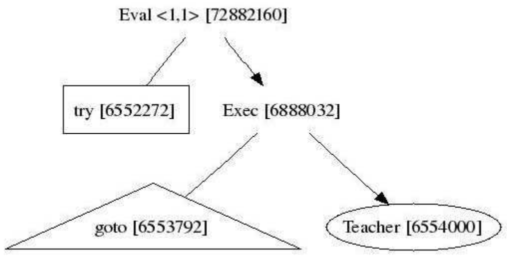


Figure 13.

and

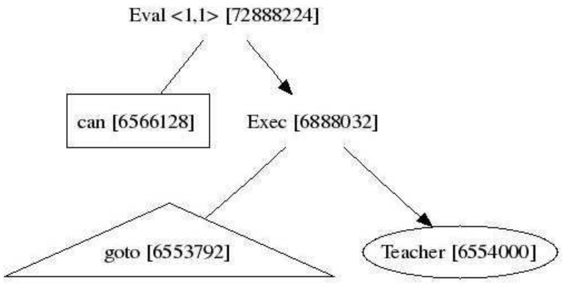


Figure 14.

which were both axiomatic.

QED!

For illustration, we finally present here an example of a plan the agent formed during the partial reward stage #1.

The inference target was:

```
EvaluationLink <0.83, 0.006>       [104296656]
  Reward:PredicateNode <1, 0>  [6565264]
```

[104296656] was produced by applying ModusPonensRule to:

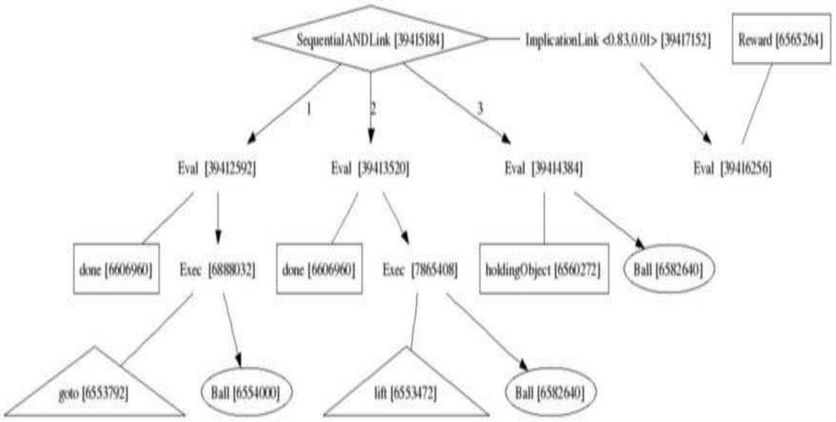


Figure 15.

On the other hand,

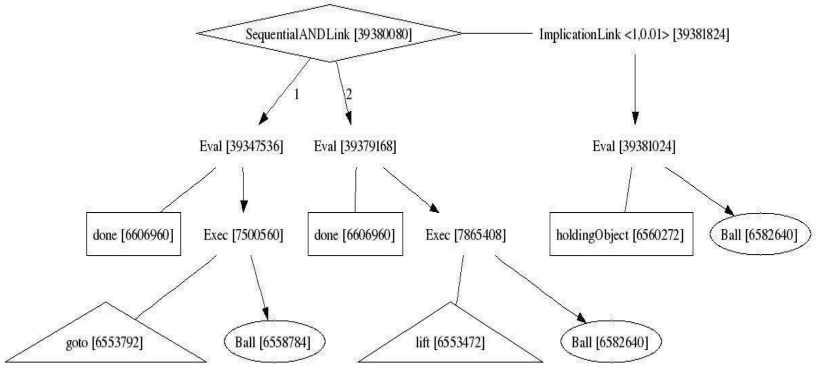


Figure 16.

This causes the system to come up with the following plan which works but contains obvious redundancy:

```
ExecutionLink <0,0.00062> [6888032]
     "goto":GroundedSchemaNode <0,0.00062> [6553792]
     "Ball":ConceptNode[typeId=10] <0,0.00062> [6554000]
ExecutionLink <0,0.00062> [7865408]
     "lift":GroundedSchemaNode <0,0.00062> [6553472]
     "Ball":ConceptNode <0,0.00062> [6582640]
ExecutionLink <0,0.00062> [7500560]
     "goto":GroundedSchemaNode <0,0.00062> [6553792]
     "Ball":ConceptNode[typeId=10] <0,0.00062> [6558784]
ExecutionLink <0,0.00062> [7865408]
     "lift":GroundedSchemaNode <0,0.00062> [6553472]
     "Ball":ConceptNode <0,0.00062> [6582640]
```

This redundancy may then be automatically removed by a simple reduction process. However, it is interesting that the said redundancy is not present in the previously described final plan, because the pattern miner was later able to find the more compact way to predict the occurrence of a Reward. So an explicit reduction step is not even necessary as the system eventually removes the redundancy on its own.

Summary and Conclusions

In this paper we have given a relatively detailed treatment of an extremely simple learning experiment – learning to play fetch – conducted with the Novamente AGI system in the AGISim simulation world. In this conclusion we will discuss some of the general lessons we learned in the course of doing this work, particularly regarding the relationship between AGI and narrow AI; and we will reflect on some ways that this particular work reflects the general developmental strategy being taken within the Novamente AGI project.

Firstly, one interesting point to mention is the relatively high level of complexity and effort required to get the Novamente AGI architecture up to the point where learning to play fetch was possible. The vast majority of the ideas that went into the process of getting the Novamente system to learn to play fetch were omitted here due to space constraints! For instance, there are various subtleties related to the use of temporal knowledge within PLN backward chaining, to the execution of schemata pertaining to ongoing actions such as "holding" in AGISim, and so forth. The reason this is worth of mention is simply to illustrate the complexity required under the hood in order for learning of even very simple tasks to occur in an "AGI-friendly" way. Of course, it would be very simple to write a software program that "learned to play fetch," if the task were represented for the program in a sufficently direct way. The machine learning problem underlying fetch is an exceedingly simple one.

The essential point is that Novamente's learning to play fetch was not completely trivial because our approach was to first build an AGI architecture we believe to be capable of general learning, and only then apply it to the fetch test, while making minimal parameter adjustment to the specifics of the learning problem. This means that, in learning to play fetch, the system has to deal with perception, action and cognition modules that are not fetch-specific, but are rather intended to be powerful enough to deal with a wide variety of learning tasks corresponding to the full range of levels of cognitive development. Making all this “general infrastructure” work together to yield a simple behavior like fetch is a lot of work – the infrastructure doesn’t increase the basic computational complexity of learning to play fetch, beyond what would be there in a simple fetch-specific learning system, but it adds a lot of “constant overhead.”

Ultimately, in a problem this simple, the general-intelligence infrastructure doesn’t add much. For instance, the PLN system is capable of powerful analogical reasoning, which means that once the system has learned to play fetch, it will be more easily able to learn to play other similar games afterwards. This capability will allow it to more easily carry out other tasks based on what it learned via learning to play fetch (a topic for a future paper). But in terms of the fetch task in isolation, PLN’s capability for analogical inference doesn’t help, and the fact of using a complex inference engine with so many capabilities merely complicates things. This paper marks merely the beginning of a series of publications involving more and more complex perception-cognition-action experiments we will undertake with this infrastructure.

Extending the scope of the discussion a little bit, these observations relate to some general differences between narrow AI and AGI work, which we believe are partly responsible for the relatively slow pace of the latter in the history of AI. For nearly any sufficiently narrowly-defined task, there is going to be some simplified, specialized approach that works reasonably well and is a lot easier to deal with than trying to apply a general-purpose AGI architecture to the problem. However, experience shows that taking specialized approaches to narrowly-defined problems yields minimal progress toward AGI. There are some problems, such as natural language conversation and autonomous scientific hypothesis, for which specialized approaches have proved almost totally ineffective – and work on making specialized systems to approach apparently-related tasks (e.g. text search, data mining) appears to help with these “AGI hard problems” hardly at all. Our hypothesis is that to approach these AGI-hard problems, the right approach is to construct an AGI architecture that in principle appears capable of solving these hard problems – and then, in order to test, tune and refine the AGI architecture, apply it to simpler problems related to the hard problems, *not worrying about the fact that the simpler problems may in fact be more easily solvable using narrowly specialized means*. That is the motivation for the work on fetch, object permanence, easter-egg hunting and other simple embodied-learning tasks currently being carried out with the Novamente AI Engine.

It is easy to draw an relevant analogy between learning in baby humans and baby animals. Puppies learn to play fetch more easily than human babies – probably, in part, because they are dealing with a less powerfully generalizable learning capability. The human baby has to tune a bigger, more general learning machine to the “fetch” task – but once it has done so, it is much better able to generalize this knowledge to other tasks, and build on it indirectly via inferential and other cognitive processes.

Acknowledgements

Thanks are due to those who helped with AGISim (including Sanjay Padmane, Teemu Keinonen, and others) and Novamente in its various aspects (including Moshe Looks, Thiago Maia and others), and earlier versions of PLN (mainly Guilherme Lamacie, the late Jeff Pressing, and Pei Wang).

References

[1] Guha, R. V. and Lenat, D. B. (1990). Cyc: A midterm report. AI Magazine, 11(3).
[2] Varelna, F., Thompson E. and Rosch, E. (1993). The Embodied Mind. The MIT Press.
[3] Harnad, S. (1990) The Symbol Grounding Problem. Physica D 42: pp. 335–346.
[4] Goertzel, Ben, Ari Heljakka, Stephan Vladimir Bugaj, Cassio Pennachin, Moshe Looks (2006). Exploring Android Developmental Psychology in a Simulation World, Symposium "Toward Social Mechanisms of Android Science", Proceedings of ICCS/CogSci 2006, Vancouver.
[5] Santore, J. and Shapiro, S. C (2003). Crystal cassie: Use of a 3-d gaming environment for a cognitive agent. In Papers of the IJCAI 2003.
[6] Goertzel, B. (2006). A Probabilistic Event Calculus. (An unpublished technical report.) http://www.goertzel.org/new_research/ProbabilisticEventCalculus.pdf.
[7] Goertzel, B., Looks, M., Heljakka, A. & Pennachin, C. (2006). "Toward a Pragmatic Understanding of the Cognitive Underpinnings of Symbol Grounding", Semiotics and Intelligent Systems Development, Ricardo Gudwin & João Queiroz, Eds., 2006.
[8] Inhelder, B. and J. Piaget (1958). The Growth of Logical Thinking from Childhood to Adolescence. New York: Basic Books.
[9] Shultz, T. (2003). Computational Developmental Psychology. MIT Press.
[10] Goertzel, B. and Bugaj, S. V. (2006). Stages of Cognitive Development in Uncertain AI Systems, this volume
[11] Thelen and Smith (1994) A Dynamic Systems Approach to the Development of Cognition and Action, Cambridge, Mass.: MIT Press.
[12] Adelson, E. and Fraiberg, S. (1974), Gross motor development in infants blind from birth, Child Development, 45, 114–126.
[13] Hawkins, J. and Blakeslee, S. (2004). On Intelligence. Times Books.
[14] Maes, P. (1991), A Bottom-up mechanism for Behavior Selection in an Artificial Creature, Proceedings of the first International Conference on Simulation of Adaptive Behavior, Meyer J.A. and Wilson S. (eds.). MIT Press.
[15] Franklin, S. (2006). A Foundational Architecture for Artificial General Intelligence, this volume.
[16] Ikle', M., Goertzel, B. and Goertzel, I. (2006). Quantifying Weight of Evidence in Uncertain Inference via Hybridizing Confidence Intervals and Imprecise Probabilities, this volume.
[17] Miller, R. and Shanahan, M. (1999). The Event Calculus in Classical Logic – Alternative Axiomatisations. Electronic Transactions on Artificial Intelligence, Vol. 3 (1999), Section A, pp. 77–105.
[18] Ikle', M., Goertzel, B. and Goertzel, I. (2006). Quantifying Weight of Evidence in Uncertain Inference via Hybridizing Confidence Intervals and Imprecise Probabilities, this volume

Advances in Artificial General Intelligence: Concepts, Architectures and Algorithms
B. Goertzel and P. Wang (Eds.)
IOS Press, 2007
© 2007 The authors and IOS Press. All rights reserved.

How Do We More Greatly Ensure Responsible AGI?

Participants: Eiezer YUDKOWSKY, Jeff MEDINA, Dr. Karl H. PRIBRAM, Ari HELJAKKA, Dr. Hugo De GARIS (Mod: Stephan Vladimir BUGAJ)

A video version of this dialogue is available at www.agiri.org/workshop

[Stephan Bugaj]: Each of the panelists will get 3 minutes to talk about your perspective on how do we more greatly ensure responsible AGI. Ari, could you go first?

[Ari Heljakka]: AGI is potentially extremely capable of carrying out any kind of action on this planet that humans could do right now – and an infinite number of more dangerous and more beneficial actions than we can do at this point. I suppose that's the premise that we all start from. Then there is the next question - what kind of artificial general intelligence is it going to be possible to create at all? And that's something we don't actually know the answer to. My point, very briefly put, is that we cannot really answer the questions of how to ensure responsible AGI before we have more information about what kind of architectures will it actually be feasible to produce, and what sort of behavior they show in the initial stages. Here, I suppose that we will actually have initial stages, further than we are right now, but not so close, beyond or equal to human-level intelligence as to actually become dangerous.

[Jeff Medina]: I think we don't know nearly as much as we need to have the confidence that many people have, or seem to have. A lot of people haven't studied ethics formally, right? That's okay, that's a usual thing, in science. But if we care about how to more responsibly move forward with AI, we either need to do that, to some extent, so that we can speak to same sort of language that the ethicists have agreed upon, and then talk and write about it amongst ourselves, or defer to someone else who has done those studies on both sides. You certainly should not just listen to what an ethicist, who similarly has not studied AI, the technical details, has to say about it. The same way you wouldn't care what a bioethicist, a self proclaimed bioethicist says, if they don't know much about actual biology. I think a lot of the theoretical information that is relevant to ethical questions has not been done yet. I think that the more responsible thing to do depends to a large extent on whether we assign a high probability to something like a hard take-off, which is where the advent of human-level AI leads quickly to superintelligence.

[Eiezer Yudkowsky]: I'm afraid I can't take refuge in claiming that I am completely ignorant of the subject. I have been studying it for the past 6 years, and if I hadn't come to any conclusions by now there would be a pretty strong question as to whether I was ever going to come to any conclusions. Ignorance can be a dangerous thing; it sometimes lets you think things that you would have to relinquish if you knew more about it. Alright, so summarize six years. There is no predetermined makeup of an AI. The space of possible AI's is vastly larger than the space of human beings. We have two problems. We have what I call the technical problem and the philosophical problem. The technical problem is if you know what you are looking for, how do you reach into mind design space and pull out an AI such that it does nice things like develop medical technologies to cure cancer (or just do it directly with nanotech, depending on how smart it is) and doesn't do comparatively awful things, like wiping out the human species. The philosophical side is: what kind of AI do you want to pull out? Even if you had all the technical knowledge to do exactly what you wanted to do, you still could have done the wrong thing. Wrong according to who? Well if you wipe out the whole human species, that was wrong according to me. If anyone would like to wipe out the rest of the human species, please raise your hand. Okay, you can make progress on this problem by walking off the cliff. Having addressed this problem for a while, I concluded that you should do the technical side first. The reason being, you don't even have a language to talk about the philosophical side of the problem until you solve the technical side. You don't know what your options are. You don't have a language to describe what it is you really want. When people first approach the problem they tend to assume that the AI is going to be like them, so they model the AI by putting themselves in its shoes. Works great if you are dealing with another human, like our ancestors for last 100,000 years. But an AI does not have the similarity to your brain architecture that another human does. If you punch a human in the nose, he'll punch back. That's a conditioned response and requires a lot of evolution to get that conditional response. If you are nice to a human, they might be nice back. Having this being an unconditional response would actually be simpler than having it being a conditional response. In other words, you can pull stuff out of mind design space that is so weird, relative to a human, that Greg Egan would spit out his gum. (That's a science fiction author who writes strange stories for those of you who didn't catch the reference.) Point is, there are really strange things in mind design space. There are things that are nicer than you imagine, and there are things that are nicer than you *can* imagine, and its one of those that you want to pull out of mind design space.

[Hugo de Garis]: Definitions, *to ensure* and *responsible*. I assume by responsible you mean human friendly. As machines approach human-level, possibly we can ensure this, if we can program in such a way that is human friendly. But I have enormous question marks about our ability to do that when they become generally smart; and of course when they are hugely smarter than us, that's a different ball game. I am obsessed by this. I see this issue, *can we do this? Should we do this?* As the issues that'll dominate our global politics. I've come to a very gloomy conclusion; there will probably be a major war on this issue in the second half of this century. I just don't think that it is possible that when machines are really smart, its possible to ensure that they stay friendly to us. I make, in my book and in the media and so forth, what I think is a very simple analogy that goes like this [*Hugo dramatically slaps his wrist and then flips off an imaginary mosquito… signifying the insignificance*]. The physicist in me says, you can take a single neuron and what's it doing? It's processing a few bits per second, you can do that with a few atoms using quantum computing techniques, right? So, the potential of the physics of computation to do what biology is doing today is just hugely superior. The potential of these machines in 50 whatever years from now, is just so vastly greater than what we are, that why should they care? So I see *the issue* that's going to dominate humanity this coming century is, do we, humanity, do we build these things or not? Now when I was putting up my hand sort of half, but half jokingly [earlier, when Eliezer Yudkowky asked if anyone wanted to destroy the human race], what the joke meant was that I think humanity has got a choice that involves facing the risk of its own extermination, if we choose to create AI's more powerful than ourselves and these creatures decide that we are a pest. They would be almost gods, right? A trillion trillion times our capacities and virtually unlimited memories, immortal, going anywhere, changing their shape. I see a kind of new religion being formed, based on this kind of stuff. Let's build these things! To describe peoples' attitudes toward these choices we need labels -- if you are pro, I call you a Cosmist: you're looking at the big picture, so-called, in the cosmos. The other group, opposed to creating powerful AI's, is called Terrans, because that's sort of their perspective, the human scale of things. So which is the greater moral evil? Risking the extinction of the human species or a kind of Deicide by refusing to build them, to build these God like creatures. If you are a fanatical Cosmist, you would say - what's one Artilect (an super AGI is called an Artilect) worth? How many human beings would you sacrifice to capture the hill? You're a general, right? 10,000? First World War, French General, you know over the top, 30% lost in no man's land, another 60% lost in capturing the trenches on the other side. What a great victory. Only 90% casualties, right? Which is the greater moral tragedy, running the risk of seeing humanity wiped out, or not building these god-like immortal creatures?

[Eliezer Yudkowsky]: I believe you are presenting rather a large number of false dichotomies wrapped into the two terms, Terrans and Cosmists. For one thing, how many lives is an Artilect worth? Implementation dependent. Let me look at its source code and I'll tell you how many lives it's worth. Is it likely to squish us when its gets big enough? Implementation dependent. Let me write the source code and I will try to guarantee you that it wont squish you. Actually looking at the source code and determining that for an arbitrary processes, is likely to be well beyond me. In theory, it's knowable beyond me because of Rice's theorem, but that's a separate issue. You are saying, it's enormously smarter, and why would it care? Because I built it. Because I reached into mind design space and selected a point in that mind design space such that I could prove that as it rewrote its own source code, it was going to keep the same optimization target. That it was going to keep on trying to steer the future into the same regions. The problems you have cited are technically addressable. We don't need a war that kills billions of people. You should be happy about that.

[Hugo de Garis]: Two major points. Technology dependent. Okay, you should do a flow chart. You should consider the various contingencies. As I see it, the keyword in this whole debate, whether you should build these or not is *risk*. We just don't know, and if they get smart enough, common sense says we don't know what they're going to do. You know we build smarter creatures all the time, with our children, who may end up growing smarter than we are, and they may turn against us. It's a possibility. Just because we build them, that's not an argument to say that they won't turn against us. If they are hugely smarter than us, you may get the Matrix scenario: you know, "You are a disease".

[Eliezer Yudkowsky]: I have a PowerPoint presentation where I actually take that scene and I show that Agent Smith is showing the humanly universal expression for disgust. This is supposedly an artificial intelligence. How does he get the brain wiring such that when he feels disgusted, his face, which isn't even supposed to be a real face, contorts into the exact expression that all known human cultures use for disgust? Common sense is not reliable here. This doesn't run on analogies and metaphors. If you want it to actually work it has to run on math.

[Ari Heljakka]: I think there's one important distinction which should be made, and that is the distinction between AGIs being under our control and, on the other hand, being friendly to humans, because the latter is very vague, and I am not sure we all agree what it's supposed to mean. I think Hugo de Garis advocates the idea that humans, as such, are not important but something more is actually what we want; and there is the concept of transhumanism, which is about humans becoming something else. So, then the question becomes, do we actually prefer an AGI system to help people change into something else, or do we prefer it to conserve the current state of humanity? That's a much more difficult question. Which question should we ask: are we able to control the AI, or what are we going to do with it?

[Sam S. Adams]: I think one answer to the question is an old line to a song, *teach your children well*, because as technologists, which fundamentally most of us are, we create tools. When we talk about making something for humanity, or controllable by humanity, humanity is a very broad space, with a whole lot of different opinions about what's right and wrong, what's good and bad, what's beneficial and what's friendly, and what's not. If I invent a new shovel, yeah it helps everybody dig holes easier, but someone also figures out that it's a pretty good machete and takes someone's arm off in the Congo. I get questions like this all the time, people come up and say - *wow, what about this thing when we build it?* They ask the moral question - *should you turn it off?* They also ask – *when it does something bad to me, who will be responsible?* The thing is, if we believe that we as technologists will control the destiny of our creations, we are fooling ourselves because we have never done that ever, ever. Okay, will mankind be able to control it? Ask yourself about any other technology that's ever been created. It is used for both good and evil as defined by individuals.

[Jeff Medina]: So in that question/comment you explicitly spoke on behalf of technologists as tool makers. AGI posses a somewhat different problem in the sense that you aren't creating something that is analogous in that tool sense. Rather it's how people use it. But, if right now you found out I am Ben Goertzel's first successful Novamente AGI, you wouldn't say - *how can I use you*? I would say - *you can't use me! I get what I want, right? They succeeded. I am human-level. I am not a tool for you to use.* The comment that we are never going to be able to control other people's use of our technologies, using a hammer to build a house or smashing skulls, doesn't seem to apply to AGI because it doesn't matter how you want to use the tool, it's what the tool *itself* is thinking about *itself*, if it reaches that point.

[Sam S. Adams]: If it reaches that point, I think we're near Hugo de Garis space.

[Hugo de Garis]: [To Eliezer.] This is a critical question. Are you saying that we, with our finite intelligence level X, that we can change the structure of this super creature such that it always remains friendly to us? Is that what you are saying is possible?

[Eliezer Yudkowsky]: Correct.

[Jeff Medina]: Right.

[Hugo de Garis]: Wow!

[Jeff Medina]: There is even among the software engineers formal verification.

[Karl Pribram]: I think what you're doing is thinking about AI, AGI, in terms of a weapon. If that's the case, we have whole history of what has happened to weapons. And somebody else is going to put up a Maginot Line which doesn't work. And somebody else is going to create something else, and we are going to have a number of AGI systems created by different parties, like happened with the atom bomb, a kind of equilibrium. I think these things will stabilize.

[Ari Heljakka]: I am afraid I am going to be awfully dull here with my idea, but if we try to approach these questions, which have to do with how do we make sure that the AGI does this or that in the future, then our discussions sounds an awful lot like the historical idealistic philosophers before the age of empirical science. The easy solution to that would be to go a bit further and actually do experiments, and make observations on the behavior, for something which is like a seed AI. The only counter argument as to why we should not do that seems to be the assumption that we'll have an extremely fast take-off, so that we won't somehow have time to make these types of experimentations.

[Eliezer Yudkowsky]: Or, a slow enough take-off that the AGI is smart enough to conceal itself.

[Ari Heljakka]: Sure. But I am certain that there are fairly long states of research where we can just use our common sense, and we'll know that the AI is not quite there yet. I know that Novamente is not quite there yet. There is so much more that we can learn before we come close to creating AGI.

[Hugo de Garis]: I think we should be expecting this now. This is so important.

[Cassio Pennachin]: I have a meta question as it relates to AI as a weapon and AI taking over the world. There is some fairly strong biological evidence that our quest for power has evolutionary reasons, which means that I don't think it's a good assumption to make that an AI will have the same lust for power.

[Hugo de Garis]: How can you be sure?

[Cassio Pennachin]: I'm not sure of anything, I'm just saying that lots of people seem to be assuming that its going to take over the world, that it's a weapon, and I'm challenging that assumption. I'm not going to assume that evolutionary bias is carried over into AI's, even if the AI is achieved through brain emulation.

[Bill Redeen]: I do think we have to assume this is inevitable… the evolution and emergence of AGI.

[Josh S. Hall]: I think it's worth thinking about what happens if a group the size of Novamente can create an AGI and it works. Or, what if Hugo de Garis creates an AGI that works. Or, what if Sam S. Adams creates an AGI that works. If that is the case, there are going to be a billion of them in 10 years. A take-off may not be nearly as hard as you think. If the take-off is going to be soft, you can't start out with the notion that your AGI is going to take over the world, because that will get all the other groups riled up.

[Jeff Medina]: A team of people with a 120 level IQ can defeat an enemy with an IQ of a 160. There is a point where we are smarter, and the enemy is defeatable. If the take-off is slow enough, you really can have an impact.

[Ben Goertzel rephrasing a question posed by Izabela Goertzel]: Isn't it likely that while you're sitting there trying to prove that your AI design is going to be Friendly, during the 48 years it's going to take you to provide the proof, someone will annihilate the world with engineered biopathogens, or with an evil, unfriendly AI or something?

[Eliezer Yudkowsky]: This is what I call the Ben Goertzel problem. [audience laugh]

[Ben Goertzel]: You need to estimate the odds that your AI is going to be friendly, and also take into account the odds of the world being destroyed by some other means while you're working on the problem. Why do you think there are reasonable odds that you would make a provable safe AI before a hundred other teams would make a highly intelligent AI? – their rate of progress should be faster because they're not stopping to bother with a friendliness proof.

[Eliezer Yudkowsky]: Presuming that my hypotheses are correct, there is a limit to how much you can understand the nature of intelligence and not notice that the AI you're building is going to wipe out the human species. In other words, there is an upper bound on the competence of the teams who are trying build the AI without understanding it properly. If they get too smart they are going to notice that they have to start over and work out deterministic designs. That's the only answer I can think of.

[Ben Goertzel]: So you're saying that all the other teams working on AI, other than yourself, are not smart enough because they should be proving theorems instead? [audience laugh]

[Eliezer Yudkowsky]: I didn't say they weren't smart enough, but I do think that if a team is seriously saying that "our AI is going to take over the world, and we don't need any assurance it's going to be friendly," I think that they have not reached the state of understanding that you get after working on this problem for a few years.

[Ari Heljakka]: I'm still thinking about the previous question, about how we actually need to do real concrete research. A relevant question is to what extent can we continue developing a specific architecture, for example Novamente, and still have a feeling that we completely understand what it's doing. Because, for example, with neural networks, you learn quickly that there is a point beyond which you don't understand the system anymore. So that is one area where you can do very concrete, fairly straightforward experimentation.

[Eliezer Yudkowsky]: I think that if you want to improve human rationality you go off and you become a Bayesian, and you go off and you join the field of heuristics and biases, which I think is already telling us a whole lot more than we are likely to get out of watching young AI's. I don't think that a human who looks at an AI are going to become smarter than the smartest humans that exist today. Maybe they'll gain something like two effective IQ points, and that'll be it. I do want to take a moment and rephrase what I call the Ben Goertzel problem. It's not specific to Ben Goertzel, but I think that maybe a better answer to the questions is that you have to hope that the first team that builds an AI is also smart enough to make it friendly. That's the main source for hope, that beyond some point you notice that you need to make it friendly and that the smartest team out there is doing that.

[Ari Heljakka]: Briefly, I do think that this is again speculation. I think it's very difficult to say beforehand how much we can learn by just looking at it, looking at the system when it is running. It's not as simple as this. We can also devise data mining methods to assist us in further understanding the system. This will provide us with empirical information. I'm not questioning the relevance of this question that we are talking about, but I am just saying that I feel very uncomfortable with the level of speculation here. I still find it kind of fun. But I definitely want to stress that doing these empirical experiments will provide so much insight that I'd rather be designing these experiments right now than speculating a lot about these questions.

[Sam S. Adams]: Question. You talked earlier about designing rules [for ensuring beneficialness of AGIs]. Recapitulating Asimov's three laws of robotics. I'm saying they are in a similar vein. The problem with these systems, is that humans are incredibly bad at writing perfect rules. Now the question I had is, I think there is a safety valve. I think there is a way to build these things and to prevent them from doing horrible things, or at least to build a pretty good safety valve, if not perfect. And someone said yesterday that science fiction was a form of experimentation of imagination into these alternative worlds. It is such an active genre -- you know, that supercomputerish AI thing that runs amuck and takes over, right? How many things have been written, or in movies, that talk about that. Now the big mistake that always happens in all of these is that the human is taken out of the loop. There is a reason why our project is actually named Joshua Hal Forbin. Because, if you are serious about building these things, you had better keep in mind what happens if you think it's smart enough to take a human out of the loop, because we as programmers suck, we write buggy code.

[Audience]: What happens if the human is taken out of the loop?

[Sam S. Adams]: Well, that's what happened with Colossus, with the Forbin Project.

[Eliezer Yudkowsky]: Logical fallacy from generalizing based on fictional evidence.

[**Sam S. Adams**]: But the point is, we have lots of people who have explored this space, have explored these arguments, without the technicals saying we are going to build one next year. But my question to the panel is, okay, so if we took a rule like - *don't give them any kind of lethal capability without a human check on that capability*, is that a useful thing?

[Hugo de Garis]: That kind of reasoning is valid when the machines are at some sort of human-level competence. But when they are hugely smarter than us, why would they continue to respect humans?

[Sam S. Adams]: It's the *pull the plug* problem, which is dealt with in the Forbin Project. Because he says – I have my finger on the button… you plug me back in or I will drop the bomb. The thing is, as soon as you take man out of the loop, whatever this thing is you create, it has a way to coerce it into do its bidding. Then you can't unplug it. That's what I'm saying. Don't go that far. Put a wall there.

Advances in Artificial General Intelligence: Concepts, Architectures and Algorithms
B. Goertzel and P. Wang (Eds.)
IOS Press, 2007

© 2007 The authors and IOS Press. All rights reserved.

Panel Discussion: What are the bottlenecks, and how soon to AGI?

Participants: Dr. Stan FRANKLIN, Dr. Hugo DE GARIS, Dr. Sam S. ADAMS, Dr. Eric BAUM, Dr. Pei WANG, Steve GRAND, Dr. Ben GOERTZEL (Moderator: Dr. Phil GOETZ)

A video version of this dialogue is available at www.agiri.org/workshop

[Ben Goertzel]: In general, what was intended when forming the question for this discussion was - what is the feasibility of achieving AGI within a reasonable time-frame? I know we have some widely differing opinions on this. I guess we can go around and have everyone give a brief statement of their views on the topic. Our moderator will be Dr. Phil Goetz, and I would like to encourage people to allow the moderator to moderate as he is a very moderate guy.

[Phil Goetz]: Everyone will give an opening statement and then we will follow with audience questions. Ben would you like to start?

[Ben Goertzel]: In terms of what the bottlenecks are, there are two sorts of answers, and the first answer is: Both generically, and for the Novamente project in particular, the biggest bottleneck is just funding into the field. It’s a common theme for many people here. For Novamente, as for most of the other AGI projects here, the main obstacle to more rapid progress is financial resources to pay people to do the work. That’s very much the case for Novamente. If we had a fairly modest amount of funding we could accelerate our progress by an order of magnitude. Next, in terms of the technical bottlenecks, that gets into deeper questions. I think the nature of current programming languages is something of a pain as it slows things down a lot. We are using C++ because it’s scalable and it lets us mange huge amounts of memory effectively. But we could move toward AGI a lot faster if there were a nicer programming language with anywhere near the same scalability as C++. Moving on: This is not quite a bottleneck, but I would say that if the Novamente system is going to fail to achieve AGI, which I think is quite unlikely, then it would be because of a failure in the aspect of the design wherein the different parts of the system all interact with each other dynamically, to stop each other from coming to horrible combinatorial explosions. A difficult thing is that AI is all about emergence and synergy, so that in order to really test your system, you have test all the parts, put them together in combination, and look at the emergence effects. And that’s actually hard. The most basic bottleneck is that you are building an emergent system that has to be understood and tested as a whole, rather than a system that can be implemented and tested piece by piece.

So to sum up: Once the funding bottleneck is solved, then you run against the difficulty of the problem, which I believe we have solved with Novamente design. But AGI is an empirical science and engineering discipline, so I can’t yet say that for sure.. You never know until you’ve implemented and tested your system in its complete form. In terms of the amount of time until AGI is completed, of course that's conditional on boring things like the level of funding and whether everyone on the team is hit by a truck and so forth. I would say, if we receive a reasonably adequate level of funding, the amount of time to thoroughly validate or refute the hypothesis that Novamente will lead to a human-level AGI, if I want to give a very wide bound, is not going to be more than 20 to 25 years, and no less that three years. I would say its highly viable to do it within 10 years, possible to do it within five years. I would also say estimating these things is almost impossible so none of these numbers should be taken very seriously. We can't even estimate how long it will take to make a word processor or a new version of Windows. The most important thing is not time estimates, the most important thing is whether you have a design for software that actually is capable of achieving very high levels of intelligence when properly implemented and tested. I believe that we have that, and I don't think we have the only possible such design. I think there are many possible ways to do it.

[Pei Wang]: First I'll talk about my own research product. The general bottleneck I think is the time I can spend on the project. So clearly I agree with Ben Goertzel with his plans. If we can get support and more resources then we can progress faster. In my case the technical bottleneck I believe will be the inference control part. It tells about if you have limited resources in a system, how to distribute it among many activities. So that is my situation.

In general for the AI field as a whole, the main bottleneck is funding of course. Also I would like to see more people get involved in this research. We are still a very small minority in AI or Cognitive Science. Most people are working on narrowly defined stuff. A few years ago I think this kind of discussion would have been taken by many people as a joke. In the last two years we have seen a lot of change in attitude. As a whole we should try to build our community and encourage more people to do more discussion and publications. Even though we have different opinions on how to do AI, I hope we recognize something in common in this room.

Another bottleneck of the whole field of research is that we all have very different opinion about what AGI means. So ultimately I can only talk about myself.

NARS should be finished, given my current progress, within two or three years. After that, optional extensions will depend on the resources question again or if other people want to get involved.

[Eric Baum]: So I think the hard thing in building an AGI is the generalization. If you build a system, unless it is extremely special, its not going to generalize. Just adding more and more stuff on top of it doesn't help. I don't think. I think that the way we got our intelligence is that evolution built a lot of specific code that understands and exploits real structure in the world, and finding that code is very hard. So I am skeptical about most other approaches that I've heard here. I don't think they are going in the right direction. It's hard for me to understand how you could think that, if you put one inference engine and another engine together and you load in WordNet, that's going to understand anything. I think that in humans, pretty much every word corresponds to a Python module that knows how to do structure and knows how to do stuff. I am trying to construct some of these for just Sokoban, that understand in a limited domain how they deal with topology. It's pretty hard. So I don't think most of the efforts are going in the right direction from my perspective. I think if we had a Cyc like program with millions of dollars of funding for ten years, five years may be, we may produce enough stuff so that it looked pretty interesting; but the other thing is, I don't think that's intelligence. I don't think you are going to hit a Singularity and its going to take off, I think it builds and builds and will be getting better slowly for a very, long time. So I'm not worried about hitting a Singularity where it suddenly takes over the world.

[Sam Adams]: Timescale and obstacles. I will talk about obstacles first. Hardware is not an obstacle. We have more hardware now than we know what to do with. I will let you in on a little secrete. In the year 2000, Paul Horn, Senior VP of IBM Research, commissioned a study. We went off and looked at all the Moore's Law like things and the result was we don't have to worry, as developers. The problem, as a hardware company, we said, was what the heck we are going to do with all that stuff? What are people going to buy it for? There are two answers. High fidelity physics simulations, which can never get enough fidelity, and real AI. We got the stuff coming. So it's how we use it that is the big barrier and that's what we are after. Fidelity, we talked about embodied systems. Embodied system situated in environments. The fidelity of our bodies, hardware or software, is paltry and nowhere near what I think are necessary or sufficient conditions to get human-like intelligence. Just because we are dealing with systems that have at best tens of sensors instead of a few billion. We don't need a billion but we could go a lot richer. I mean the simulator system I'm working on, and AGI-SIM, they're all fairly simplistic and they need to be a whole lot richer. Training time is another big one. You start talking about developmental systems that are going to do their own learning, then you've got to figure out how the heck you are going to train them fast enough. Now I presented yesterday that we believe we have a way to do that with some engineering tricks and taking advantage of social tricks like the internet and multi-user games, but that's a real problem. If you are going to do developmental robotics or developmental intelligence, you've got to be able to say how are you going to train it fast, and that's a limitation. Shared resources and minds are a barrier. Probably in no other field, and those of you in other fields may strenuously argue with me, is there more "NIH" (not invented here) than in the computer science field. Frankly if we are all determined to keep stepping on each others toes, we won't get there until one brain can figure it all out. We all have separate projects -- even though they have similar architectures, they are incompatible, we could not fuse them if we tried. The same problem hit AI in the early days. Everyone had a system; no one could do anything with it. That's a serious problem. If we as a community want to solve the problem, that's the limitation we need to address. Funding being a limitation? I don't think so. I see this as someone who's job it is to convince people of the value of technology so they will invest. That's my job at IBM fundamentally. And the reason is you have to build something compelling to convince them of the value. As long as you are building toy problems, no one is going to care, no one is going to believe us, at least more than once. And guess what? The previous AI Summer and Winter already used up all the toy problems. None of them are interesting. Time frame? I believe that we are looking at serious contenders in 10 years. But they are going to have all kinds of interesting limitations, but people will definitely be sitting up and noticing and saying, wow we are really, if we are not there, we are awfully close. The common man will be able to say wow we are almost there. Once that happens, even get close to that, then the flood gates of money start opening, because everybody wants to own it. So then it will accelerate.

[Hugo de Garis]: I see three major approaches to producing AI, or as I call them *Artilects*. One could be a slave, maybe slave is the wrong word, a slave to neuroscience. Common sense says if you imitate the brain as closely as possible or sufficiently, you will end up with a conscious intelligent creature because we have ourselves as existence proof that its possible to put molecules together to make such creatures. So the brain mimic approach. Second one is just an engineering approach. Don't be a slave to neuroscience, just do whatever you like. And the third one, of course, is just the hybrid. Probably most people will do both. For example, I'm a little surprised at this workshop that there is not a larger contingent of the brain modeling crowd. I hope we will by next year's conference, or workshop. For example, Henry Markram, the Swiss IBM, the Blue Brain project, that kind of thing. So I hope at the next workshop there will be a bigger contingent of that aspect. Obstacles? Well you just talked about AI Winters and Summers. I don't know how many cycles it's been through. Is it three, five? Is there a lesson to be learnt there? I guess one obvious one is just ignorance. Even if you take a purely engineering approach, how the hell do you build an intelligent machine? My sense is we don't even know what the target is; we don't even know how difficult it is to do that thing. So it's very hard to figure out how long it's going to take to get there. Basically if you take the brain mimicking route, we will get there. Time scale? Well again, just my own opinion, I think humanity will have to have full knowledge of nanotech to enable us to create tools, sufficient tools, to be able to figure out how the brain works. Because we are talking – how many synapses do we have? What, a quadrillion? This is a huge tremendous problem. But with molecule scale tools, then I can imagine, let's say in the 2020's or that kind of time frame, then nanotechnology will explode; and then once the tools are there, and lets say in the 2030's and 2040's, there will be a corresponding explosion in our knowledge of the principles of how the brain works. As soon as some newer science principle is discovered, immediately it will go into the engineering, to the point where the two fields will just merge, they'll wed. The newer engineers and newer scientists will become effectively the same thing. So we'll have real AI's, let's say near human-level, second half of this century, with that logic. Now, if Ben Goertzel's claim is true – I mean, I'm open. I'm open to the idea. Maybe I'm being conservative and wrong on this. Maybe a purely engineering approach can correct the problem, and we haven't been able to do it till now because we just haven't had the hardware. Right? Moore's Law hasn't allowed it. But think about it: 10, 15 years into the future, we are in for an explosive increase. It's just the math. The size of the Moore doublings now is just huge. You know the argument about the lily pond. You have a bunch of lilies and I keep doubling their size every year. For many years it's just negligible. The size of the pond is huge. But the second to last doubling its sort of like half of the pond, and the last one is "boom", its there. Most of the increase will occur in the next few years. We will see it easily. I really agree, hardware is not the issue. We will have more hardware than we have ideas. My gut feeling, that's all it is, is that we are not going to have those ideas until we get them from neuroscience. But that might be conservative. And I'm open, you know, as the engineers; the engineering approach comes up with a way of doing it, because we have the hardware, well you know, that would be great. So okay: optimistically 10, 15 years -- or more realistically in my view, 50 plus; between 50 and 100, definitely this century. That's scary.

[Stan Franklin]: The disadvantage of being close to the end of an eight person panel is that essentially everything I wanted to say has already been said. Maybe not everything. One of the things that struck me about this workshop, as other people have said -- Ben Goertzel has said it, Sam Adams has said it -- is that there is a fair amount of commonality that I see. Maybe even more than I expected, but that's not true, because we are working on the same problem, and we are going to have the same issues. So it's not surprising that solutions look much alike in ways even though they can be very different otherwise. But it strikes me that a real drawback here is the lack of a common ontology and the lack of a commonly used vocabulary. A commonly understood vocabulary, to use in discussing things with others. I found over and over again in my interactions with individuals here that our conversations could have been twice as productive in the same amount of time if we hadn't had to spend time getting around the vocabulary issue. The ontology issue. So let me say, I've had that same experience in dealing with psychologists and with other computer scientist. I don't think this is that unique to us in any way. The psychological memory researchers still have no good vocabulary for what memory is about, and they argue with themselves. Let me not even mention the philosophers. So commonality is one thing, vocabulary ontology is another, and the third is this business of funding, which everybody has talked about, and certainly I suffer from it like everybody else, and I think it's a major bottleneck. What can we do? One of the things that I would hope that we can do that I see sort of hopefully is starting here an AGI community that can interact with some regularity and try to overcome some of these bottlenecks. Then I think we should cut this thing short enough to have some discussion this afternoon about how to do that. In any event, I think that that's a major issue – how to get this community going, because the community can do something about the vocabulary, the – and the community may even be able to do something about funding. We need to think of ways to do better PR. As Steve Grand pointed out to me before, a lot of these bottlenecks, such as the funding issue, are because we are not presenting – we and lot of others as well - aren't presenting a good enough case to the government, or maybe it's not even to the government. He pointed out that physicists get large amounts of money. If we had the money for the whole crew of us for one cyclotron, or one large telescope, it would – how long would it keep us? A decade? I don't know. I haven't done the arithmetic. Okay in terms of time frame, again as Ben Goertzel said earlier, testing parts of the system doesn't work, and in an AGI agent that won't do. We have to have full systems and test them. In my view, you are not going to be able to build an AGI agent from scratch, it's just too complex and what you are going to have to do is build in whatever you can build in and let it learn. And to test for learning things takes a long time. Look how long it takes for a human child to learn to get to the point that we would call it really intelligent. I mean it is certainly some years to do that. So the whole business of testing these agents is going to be very, very long. So I am not optimistic about the length of time that it will take. Although – yeah, I am just too scared about predictions to make any, so I think I am just not going to.

[Steve Grand]: Well, I am tempted to say that it is *all* a bottleneck. If there was a panel like this in 1960 that was asked the very opposite question: which aspects of AGI do you think are really *easy*, what answers would have been given? Those would be my answers for the really hard aspects. I'm talking about things like vision and motor control, sensory stuff and perception. Rodney Brooks has a copy of a memo from Marvin Minsky, in which he suggested charging an undergraduate for a summer project with the task of solving vision. I don't know where that undergraduate is now, but I guess he hasn't finished yet. So these are really, really hard problems. But they are the things that all of us here, or most of us, not including me, are sort of brushing under the carpet. I think since this is primarily a symbolic AI conference, the problem we have is extending symbolic AI downwards far enough. Meanwhile, bottom-up AI people have the opposite problem - extending upwards. But you have to do it. Imagine that you were born unable to see, unable to hear, unable to feel, unable to move. How smart would you be? I think you wouldn't be smart at all, and intelligence *requires* you to have these sensory motor skills and extreme competence in perception. Yet we just don't know how to do those things. And where we think we do know, they don't fit with symbolic systems. So there's a real interface problem, and that's because I don't think we know the appropriate representation yet. So that's why I say all of it is a bottleneck; we just don't know how to do it and I think we need a whole new paradigm; a new set of representations. At all levels in the hierarchy. As for "timing" -- Alan Turing famously made a prediction 50 years ago, that by the end of the century we'd have human-level intelligence. His prediction was proven wrong at the very second that computers were about to demonstrate how stupid they are, because they couldn't add one to 1999 and get 2000. So I am not going to be drawn into this and make the same mistake. Except that I think that we really do need a breakthrough, and the thing about breakthroughs is that you can't say when they are going to come. You can't say you are halfway towards a breakthrough. It doesn't make any sense. It just happens. What we need is a paradigm shift. You can see signs that there are the beginnings of a paradigm shift: particularly in the neurosciences; particularly as new technology is coming to neuroscience that enables us to see stuff we couldn't see before. So there is a hint of a shift. But when it's going to click and somebody is going to have that "aha" moment, I don't know. It could be tomorrow or it could be in 100 years time. I really couldn't guess.

[Phil Goetz]: First question for the panel. Is AI a science? I note that we don't have a single journal that all of us reads. When we do read journals, we very rarely see the case that someone frames a hypothesis and a null hypothesis and has a statistical test to distinguish between them. We don't have repeatable experiments because no one in the field can repeat the experiments of anyone else in the field really. We don't have post docs, we don't have degree programs, we don't have a curriculum, we don't even have an AI prize, that's the equivalent of the Nobel prize or something.

[Ben Goertzel]: There's the Loebner prize.

[Phil Goetz]: But nobody's won it. Do we even really have a community in the way that many others sciences do? So I guess I would like to pose the question to whoever among you would choose to answer it: Have we reached the status of being a science and if not, what can we do to be more scientific?

[Steve Grand]: We've never really been a science. AI has always been an engineering discipline. Engineering tends to draw upon science rather than contribute to it. And as for community: yeah, one of our problems in being a community is that we all hate each other's guts. We are forever dividing ourselves up, rather than coming together, which is probably a sign that we don't have a good theory yet.

[Audience]: Ben, you seem more optimistic. Could you talk about your perspective?

[Ben Goertzel]: Well I have a quite different opinion than that of Steve Grand in that I don't think an amazing conceptual breakthrough on the level of the discovery of the quantum or curved 4D space-time, or something like that, is needed to create general intelligence. It might be needed to create provably stable friendly AI, like Eliezer Yudkowsky would like. I tend to think of the brain as a complex system composed of a bunch of evolved kluges for solving particular problems, which have been hacked together and adapted by evolution. I think if you assemble subcomponents solving the appropriate set of specialized problems, as well as a fairly weak general problem solver, and they are hooked together in a knowledge representation that works for all the components, with learning mechanisms that let each component learn from each other -- then you are going to have an intelligent mind that can be taught. You make it a baby and you teach it, then it grows up. I don't believe an amazing breakthrough is needed, I just think we have not done what I have just said, in part because hardware hasn't been there, and part because saying it is a lot simpler than doing it, because what I just said involves a whole bunch of different components – each of which has to basically work, with a lot of testing and tuning of the parameters of the learning algorithms. I agree that it's an engineering problem, and it's a big and hard engineering problem, but I am not sure it requires an amazing conceptual breakthrough in order to do it. You know there is a lot of stuff in the Novamente design that I think are moderate level conceptual or scientific breakthroughs, which are really interesting stuff, as a single example, the stuff Moshe Looks is doing. I think Moshe's work shows that we have figured out how to make evolutionary learning work, which is one among many mid-sized breakthroughs we've had to make as part of creating the integrative Novamente system. But none of those things is an amazing epochal breakthrough, and I don't think one is needed.

[Steve Grand]: I wasn't actually suggesting that we needed a magical breakthrough, just a breakthrough, and it may be very simple: a change in the way we look at the problem.

[Ben Goertzel]: I don't think so. I think the brain is a mess. I think when we see and fully understand it, we will say, goddamn, I can't believe this thing actually let's us think.

[Steve Grand]: Well I disagree with that. I think only *parts* of the brain are a mess.

[Mike Vasser]:Does the problem look better now than it did 20 years ago?

[Sam Adams]: I think the empowerment of the individual scientist is so much greater today than it used to be. Used to be you'd have to have a good budget to build the big system. Now you don't. The tools we have are so powerful, and the components that we have, that a single person can sit down and build a really significantly large system. And so you have the opportunity -- it is sort of like what Steve Grand was saying, this is a problem that is best addressed today at least by having it all in one head because it is just too complicated to split up. And I think that's a real advantage we have now that we didn't have before. Just empowering of the individual mind so that one person actually can build it, or at least build a significant chunk of the system on their own. Drawing from everybody else's research and knowledge, of course, but they can implement it, they don't have a big barrier to implementation.

[Eric Baum]: I think there have been advances in knowledge over the years. It's not like we haven't learned anything. I mean the whole field of learning made a lot of progress. If we go back – over my career, you go to 1980 and you didn't understand what learning was, and now I think we have a pretty good understanding.

[Ben Goertzel]: I would echo that sentiment. The Novamente system in particular builds on a lot of insights made over the past 20 years or so. I mean the stuff that we are doing in the evolutionary learning module builds on the Bayesian Optimization Algorithm and so forth. And the probabilistic reasoning system we use is unique – but it would not have been nearly as easy to construct if not for all of the previous work in various aspects of probability theory. There has been a lot of probabilistic reasoning work in the mainstream AI community. So even though I think the traditional AI community has not focusing its efforts ideally, because of ignoring AGI, by and large, they have done plenty of useful work on various component technologies that can be used to make AGI systems.

[**Stan Franklin]**: Can I add to that a little? In my opening talk here, I had a slide listing a large number of theories from cognitive physiology that had been embodied in the LIDA system and that couldn't have been done some few years ago, some 20 years ago. It certainly seems to me that when I first started looking at this thing about 15 years, it seemed so far in the future that I though it silly, but let's do it anyway. Now it seems much more palpable than it did because I have got this cognitive theory of everything that looks as if, if you give it a chance and let it move along, it might in fact be developed into an AGI.

[Eric Baum]: To go back to your question before about the science, the one thing you said was that nothing is repeated and I think that's just false lots of things get repeated – I mean I learned here that somebody repeated Hayek [a learning system developed by Eric Baum, which was replicated by Moshe Looks], but that's a very minor one. For example chess programs are repeated you know multiple times, or support vector machines. I could give a long list.

[Phil Goetz]: I guess I was thinking of general AI systems, such as when Ben Goertzel talks about Novamente, I've not heard of anyone trying to duplicate what they report.

[Eric Baum]: Novamente, in some ways, it seems to me like the systems of the 60's, which were hard to repeat, and maybe weren't repeated and that was a criticism that used to be made, but I don't think that's true with a lot of the progress that has come about now.

[Hugo de Garis]: I'm no expert in brain scanners, but people tell me they are subject to Moore's Law. Talking to a neuroscientist, I asked, what's state of the art in brain scanner work today? She said millimeters and milliseconds. So you know its true that – well it is true that its subject to Moore's Law. Then you get into the Ray Kurzweil scenario of being able to scan the brain to the point of every synapse. So you can imagine downloading all that information into a kind of future hypercomputer, and they're coming. I talk about Avogadro machines. If you know any chemistry, you know Avogadro's number, the number of atoms you hold in your hand, which is effectively a trillion trillion, so you have all the components and hardware needed. You can imagine huge amounts of information on how the brain is structured. Then you can start doing experiments on structure and trying to figure out how it works. I see this eminently doable in the next several decades.

[Stan Franklin]: I'm concerned about brain scanners being… I think they are fast in time, but the ones that have good spatial resolution, takes seconds. I have serious doubts about how far we are we going to get with those. But I'm not an expert there.

[Steve Grand] At the end of that process, all we'd end up with is another brain and we already know how to make them. We also already know how to take them apart and we still don't understand how they work.

[Audience]: How much do you want General AI, and how afraid are you of it?

[Sam Adams]: I want it a lot, and I'm not afraid of it at all.

[Hugo de Garis]: To me it's a religion, and I'm dead scared.

[Stan Franklin]: My then 16 year old daughter, who had read Bill Joy's article, said to me, daddy these intelligent systems, that's what you do isn't it? Why do you do it? The answer I had to give her at that time, which is still my answer, is that I think we are too far away to do ourselves any harm yet, and by then, maybe we will be able to figure out someway not to do ourselves harm.

[Steve Grand]: I'm a member of Sam Adams' religion and I want it. And I'm not scared of it. Intelligence doesn't scare me at all; it's the stupid people who scare me

[Sam Adams]: Artificial stupidity is much more dangerous.

[Ben Goertzel]: My response is a little more like that of Hugo de Garis'. I want it and I'm moderately, though not massively or panicked-ly frightened over it. There is obviously the possibility of very grave dangers, yet there are also a lot of other possible dangers that are out there, from things that don't have all of the good aspects that AGI may have.

[Pei Wang]: If I say I'm scared about it, you should not believe me; otherwise I should not be working on it. I think the same thing is for everyone sitting on this panel.

[**Hugo de Garis**]: Could we take a vote?

[**Phil Goetz**]: Audience question, raise your hand if you are scared of AGI?

[About half the audience raises their hands]

[**Phil Goetz**]: Let me ask a more controversial question. How many of you, if you had built your AI, and you thought it was of human-level intelligence, and you really didn't have any means for controlling it, how many of you would turn it on? Raise your hand if you would turn it on?

[Izabela Goertzel]: Human-level is not that smart. [laughter]

[Phil Goetz]: But that's how I am phrasing the question. So that's one, two, three, that's about 10 hands out of 35, 36 people.

[Karl Pribram]: From my experience, you turn it on, and five seconds after its working, it breaks down. [laughter]

[Mike Ross]: The opportunities for collaboration seem so small. There seem to be only occasional opportunities for working together and testing out things. What does the panel have to say?

[Sam Adams]: There is a big difference between people's code and people's ideas. People tend to not want to use other people's code. Programmers, you know, love writing their own code. It's a problem software has in general, but the thing we do have a possibility for, that I've been kicking around with some of the other people here, is just seriously collaborating consultatively with each other. Let's not go away and not talk to each other for two years, and then come back and see how we proceeded. In this day and age, we ought to be meeting at least every six months, as a large group, at least, with a mailing list, a wiki, who knows what else, keeping us in tune and freely consulting. You know these things are so deep, there is no way in the world that you are going to give away the groceries on one of these systems. These are huge complex systems. I can go talk with Stan's team about LIDA, because we are similar enough, that I could probably say you missed a feedback loop there. That may save them years of work, that may be a dead-end they don't find forever, whatever, that's the kind of collaboration I think we need to have to spirit of. Granted, whoever builds this thing is going to win the Nobel Prize, but really I want to see it work, I mean is there anybody who doesn't want to see it work -- hopefully we can turn it off, right? So if Ben Goertzel does it, great, and he uses a couple of my ideas along the way, fantastic.

[Phil Goetz]: Do we need to cooperate now before those floodgates of money that you spoke about open up?

[Sam Adams]: When it finally gets greedy then, then it's over.

[Stan Franklin]: The first thing, the very thing that Ben Goertzel said, and I have already made arrangements with one or two people to collaborate in the kinds of ways that Ben is talking about and trying with others as well. Let me also say that right now I have members of the research team that's working on LIDA all over -- most of us are in Memphis, one is in Berkeley, one is in New Zealand, one is in China, two are in this area, and they all contribute one way or another. Let me say, by way of invitation, there is plenty of room for others if you want to get on board, please let me know.

[Eric Baum]: I agree that the three of you have similar systems, and also the three of you are complaining, well at least two of you, are complaining that you need a lot more funding to finish them. You might consider merging them and making one open source system. You'll have a lot more manpower and you could finish all these different pieces.

[Sam Adams]: Open source of these technologies is a real opportunity.

[Stan Franklin] That's a possibility.

[Pei Wang]: One problem, or difficulty of cooperation in this field as far as I can see here, is that we are actually not exactly working on the same problem. We are already seeing several very different paradigms in the field of AI. Also there is the problem of terminology. I am fully aware that my working definition of intelligence is quite different from many other people. But I do have the research to follow it up. I believe the same for the others. So my best suggestion is that we try to cooperate to push as far as possible. I will try my best to find overlaps so we can cooperate.

[Ben Goertzel]: I largely agree with what others have said. I do think though that, going back to Mike Ross' original question, there is some fundamental intractability in this particular domain, in terms of communication between different teams. I think it is kind of similar to how it's so hard to understand someone else's software code compared to writing your own. Even if it is well constructed code, it's often still quite painful to read and understand the details of someone else's code. Similarly, if you have three AI systems, and they are kind of similar at the high level, it will still be hard for the creator of each system to understand the other ones. You just make so many small decisions for so many complex combinations of reasons, when you build a system like this; and to communicate all these to someone new takes a really long time. If we opened up the Novamente design to someone, it would take a long time for them to understand the things we've done well enough so they can really intelligently give advise and spot something that's wrong versus right. So I am not as confident as Sam Adams that in a brief period of acquaintance someone could look at the Novamente system, or his system, and say, well you missed a feedback loop. I am afraid the things that we are missing, if we are missing things, are things at a deeper level and would take a much greater acquaintance with the system to figure them out. But having said that, although I do think there is a fundamental issue there -- which is because we are engineering complex systems based on somewhat different approaches – there is certainly the option for a lot more sharing than this workshop affords. Hypothetically if we choose to take the time -- if each of us took one day to describe the inner workings of our system to each other, then spend a day trying to get a common language, then spend another day describing our system in the common language – we could get a lot of sharing done. I am not necessarily advocating that – but if we did that, I am sure we'd have a much higher chance of understanding what each other were doing and making profitable suggestions about what each other were doing. The final point I wanted to make, is something easier than making suggestions about the core logic of each other's system or fusing each other's systems (as an aside, actually I don't think Sam Adams' and my Novamente are similar enough that we would ever agree to make them the same system), is that there are big opportunities to make useful common tools. The example that we harped on throughout this workshop is the AGI-SIM simulation world. If various teams wanted to make a simulation world, we could agree to collaborate on this -- everyone could put in a little bit of work to help with the annoying problems of making legs work right and dealing with different kinds of vision and so forth. And there are probably many other examples of shared resources that could be helpful; but that particular resource (AGI-SIM) is interesting because – not only is it open source and could be collaboratively worked on, but it could provide an environment for different people's agents to interact with each other, which is kind of an interesting thing. It'd be cool to have an online playground where AI's could just kind of cruise around; and that actually would be very helpful in attracting funding as well. If there was a public, open source AGI-SIM server, different AI teams could log on and let their AI's play together, and that might help us get some of the funding that we want.

[Stan Franklin]: In spite of all the commonality, the major difference is in how we are going represent data. It seems that any such system has to have a common currency that runs through it; we have certainly had to do that with LIDA. In terms of what Steve Grand said earlier, I think our representation can bridge the gap, I think we are somewhere in between, where we move up to the symbolic and then down. This is still mostly conceptual either going down or going up. We have gone up a little bit in the LIDA model, with natural language understanding, and that kind of thing, but not much.

[Sam Adams]: One interesting opportunity that Ben was talking about is possibly cohabitating a common avatar space to train in. The interesting possibility there for collaboration is having one system train another -- and in training each other, teach each other what they know.

[Ben Goertzel]: New possibilities for artificial stupidity. [laughter]

[Ben Goertzel]: Well I feel you have all heard more than enough of my capability to vocalize, so I'm not going to make very lengthy closing remarks here. In general, I'm pretty happy with the way this workshop has gone, I think we have gotten a bunch of really good people, a lot of interesting approaches to the general intelligence problem -- and as many others have echoed, it seems to be close to the time when AGI research is going to take off, in terms ideas and systems maturing, and in terms of the community of researchers growing, more and more new people coming into it. I have been thinking about this stuff for a long time, and working on it for a moderately long time. Others have been working on it significantly longer than me. I'm 39 and I can say that it's definitely been an AGI Winter throughout most of my 20's and 30's. I can see now that it's starting to thaw, in that we can have a workshop like this and you can actually get a bunch of people to come to it. In the last few years, AAAI and IEEE and IJCNN and so forth are having workshops and symposia now and then explicitly touching on human level AI. You can go and give a talk on an integrated AI system or a would-be AGI system and only half of the room points their finger and laughs. So, its really significant progress; and I think the community will grow and we'll see more and more interesting stuff happen. Regardless of whether my own personal optimistic time estimates come true, I feel like the exponential growth curve of AI is definitely cruising upward in a noticeable way, and its certainly going to continue to do so, which is pretty exciting. We don't have specific plans for our next workshop but in some form or another there will be more of these and we hope to draw in an even broader community. Thank you all for coming and for all your contributions!

This page intentionally left blank

Advances in Artificial General Intelligence: Concepts, Architectures and Algorithms
B. Goertzel and P. Wang (Eds.)
IOS Press, 2007
© 2007 The authors and IOS Press. All rights reserved.

Author Index